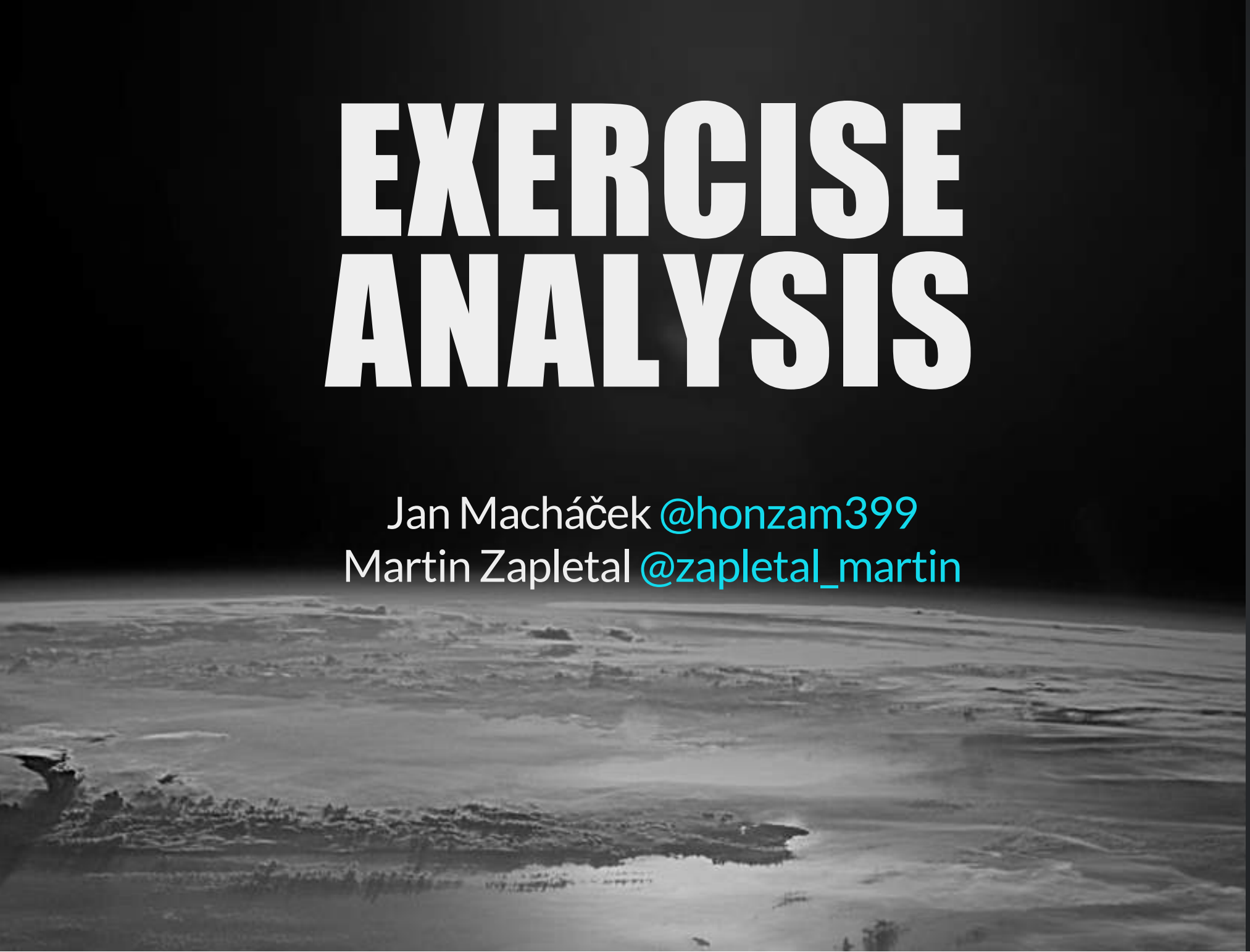
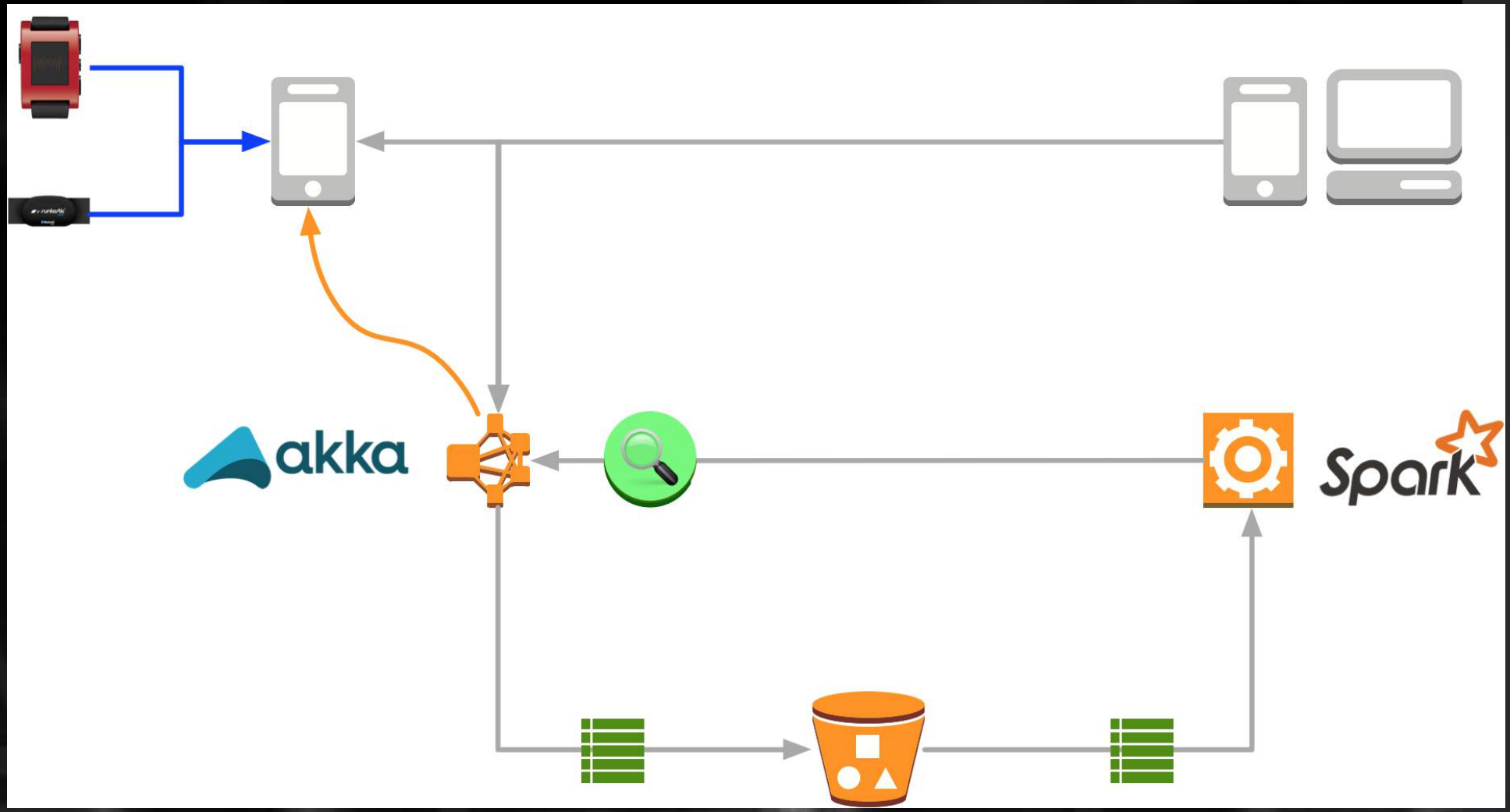
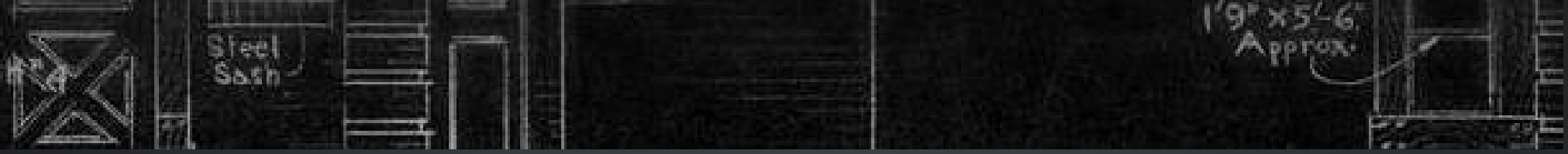
EXERCISE ANALYSIS

Jan Macháček @honzam399
Martin Zapletal @zapletal_martin



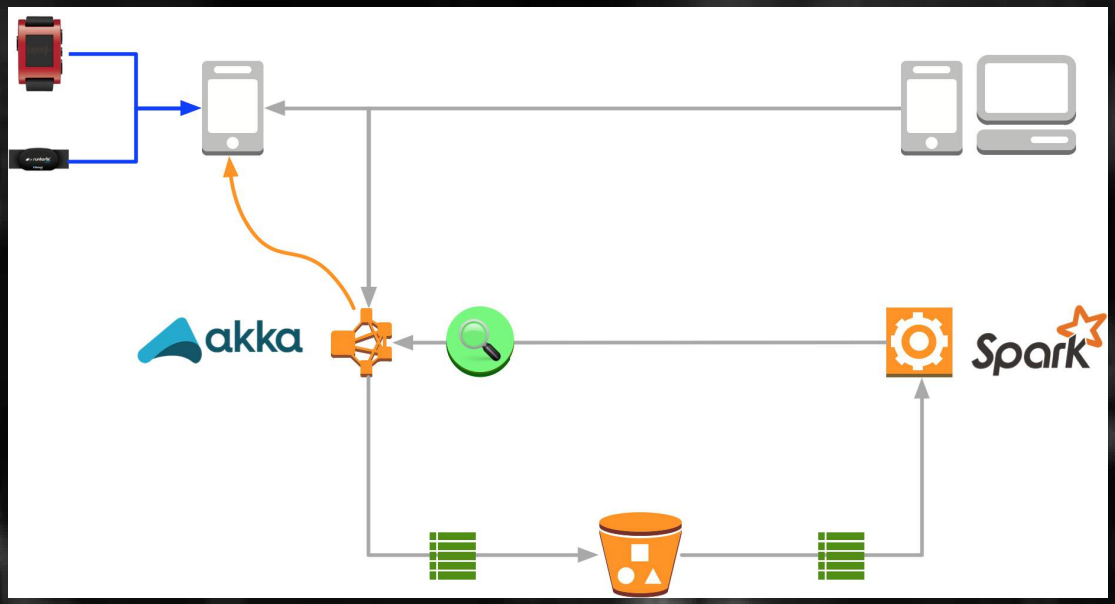

akka
Spark

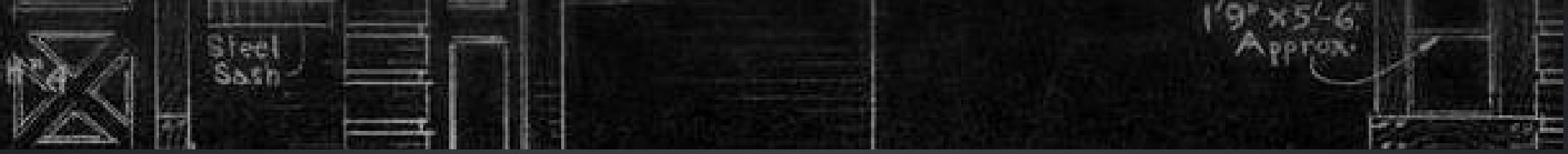

Steel
Sash
1'9" x 5'-6"
Approx.

muvr / **open-muvr**

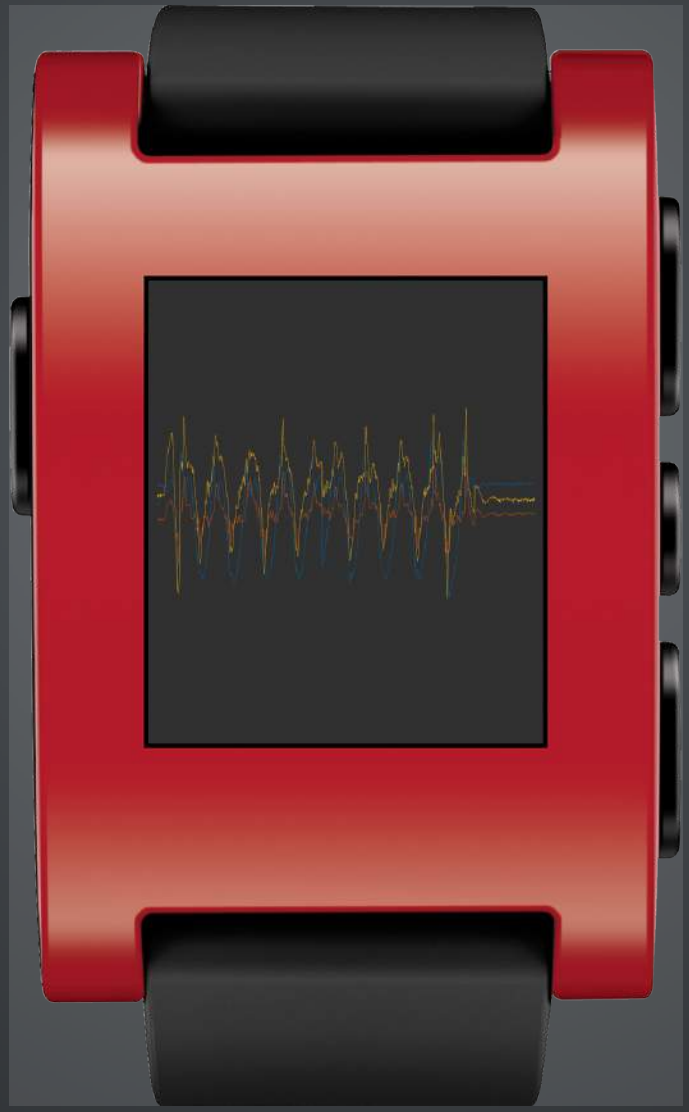
Open-source core of muvr http://muvr.io — Edit

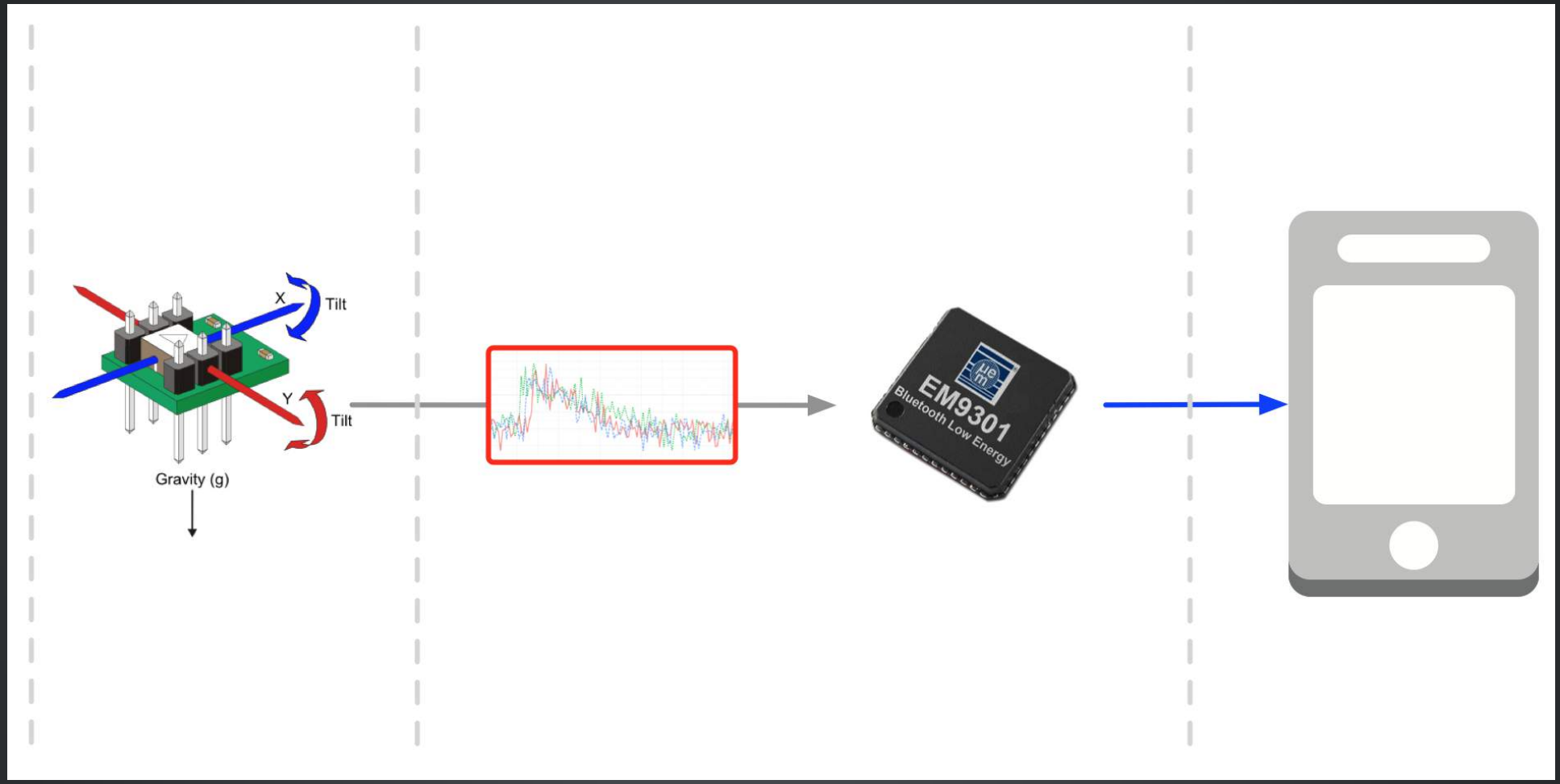
C++ 58.3% **Swift** 19.5% **Scala** 16.9% **R** 1.5% **CMake** 1.0% **C** 1.0% **Other** 1.8%

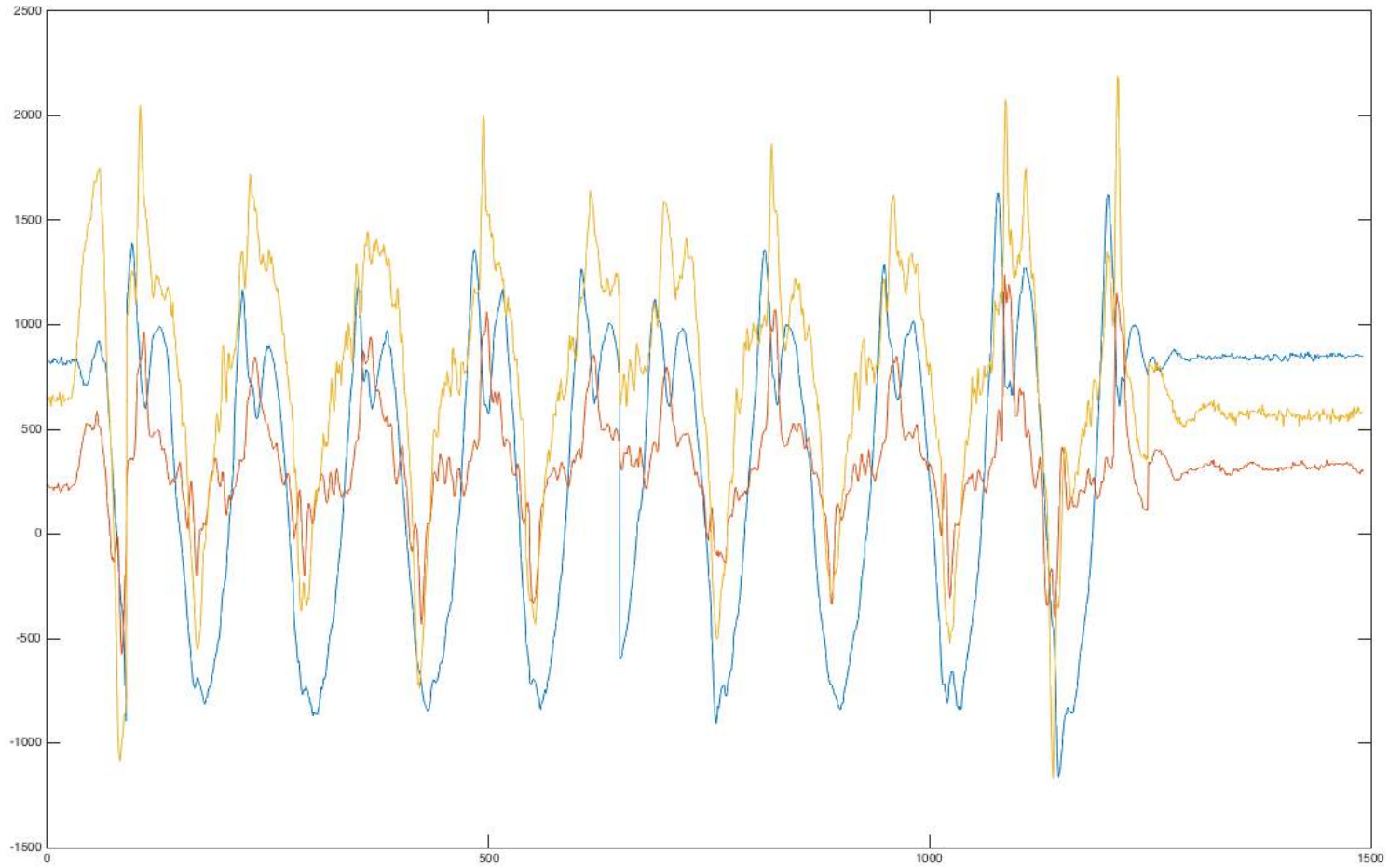



Steel
Sash
1'9" x 5'-6"
Approx.

WATCH




X
Tilt
Y
Tilt
Gravity (g)
EM9301
Bluetooth Low Energy


2500
2000
1500
1000
500
0
-500
-1000
-1500
0
500
1000
1500

ads

1491x3 double

	1	2	3	4
1	824	232	632	
2	824	224	656	
3	816	224	648	
4	816	224	608	
5	824	224	648	
6	816	216	664	
7	840	208	648	
8	832	224	640	
9	832	224	632	
10	832	216	632	
11	816	224	656	
12	816	240	640	
13	816	216	648	
14	824	208	680	
15	832	200	640	
16	848	208	608	
17	824	216	656	
18	808	224	664	
19	816	232	632	
20	816	232	640	
21	808	232	640	
22	824	240	648	
23	824	224	640	
24	824	208	648	

SANE PEBBLE APPLICATIONS

- Responsive
- Resilient

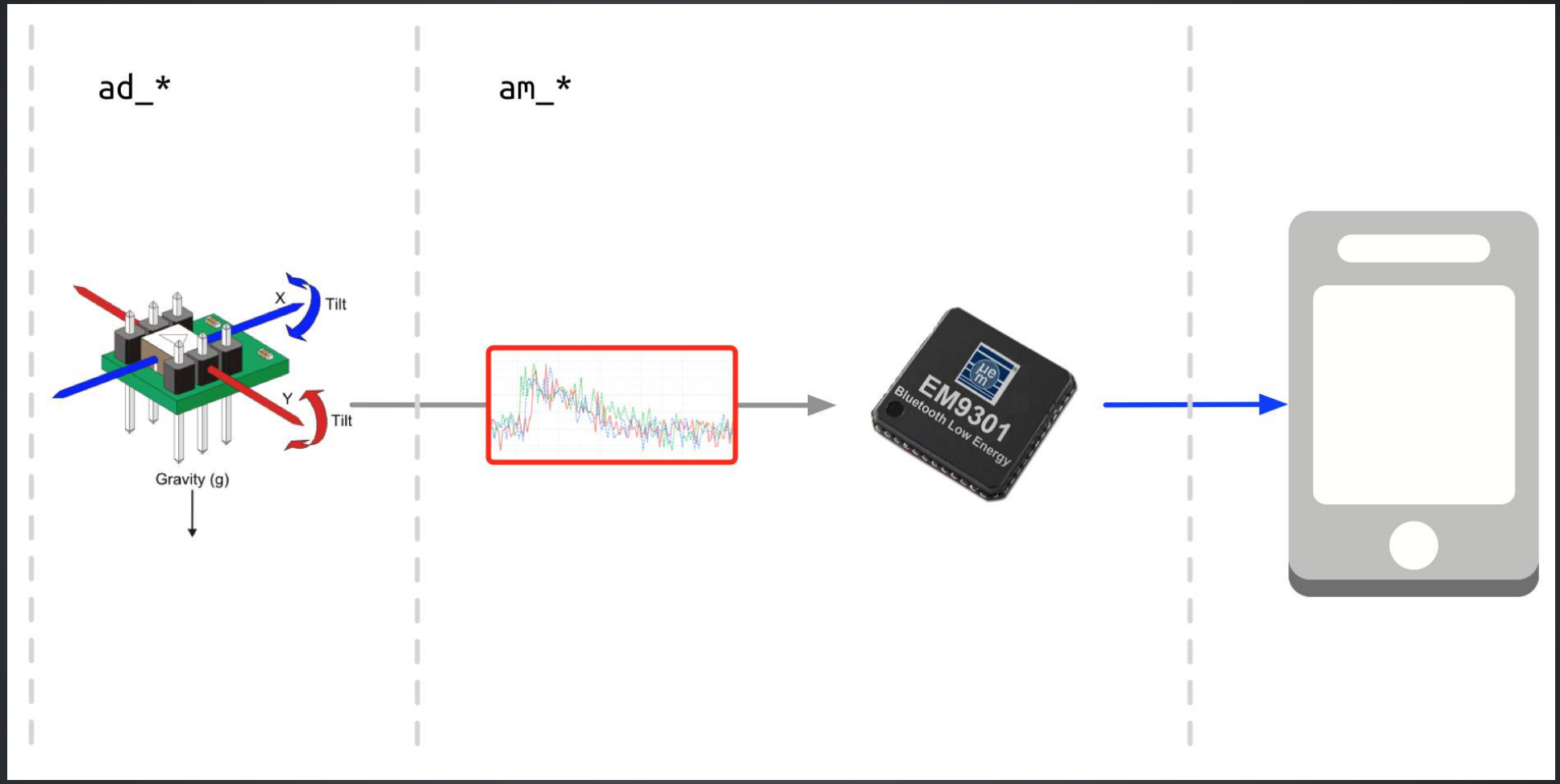
- Core components in C, tests in C++
- *Multi-platform*: buildable & testable outside Pebble
- Smallest possible Pebble-only code
- Sane development tools (e.g. CLion, emacs)

- **core**
  - **main**
    - am.c
    - am.h
    - …
    - queue.c
    - queue.h
    - CMakeLists.txt
  - **test**
    - gfs.cc
    - dl.cc
    - CMakeLists.txt
  - CMakeLists.txt
- **pebble-mock**

cmake
make
ctest

- **src**
  - main.c
- appinfo.json
- wscript

pebble build
pebble install …


ad_*
am_*
X
Tilt
Y
Tilt
Gravity (g)
EM9301
Bluetooth Low Energy

```
struct __attribute__((__packed__)) header {
    uint8_t  type;
    uint8_t  count;
    uint8_t  samples_per_second;
    uint8_t  sample_size;
    uint8_t  time_offset;
};

struct __attribute__((__packed__)) threed_data {
    int16_t x_val : 13;
    int16_t y_val : 13;
    int16_t z_val : 13;
    uint8_t valid : 1;
};
```

```
typedef void (*sample_callback_t)(uint8_t *buf, uint16_

#ifdef __cplusplus
extern "C" {
#endif

    int ad_start(sample_callback_t callback, uint8_t f
    int ad_stop();
    void ad_update_time_offset(void *header, uint8_t t

#ifdef __cplusplus
}
#endif
```

```c
#include "ad.h"
static struct { ... } context;

void accel_data_handler(AccelRawData *data, uint32_t nu
                        uint64_t timestamp) {
    ...
}

int ad_start(sample_callback_t callback, uint8_t freque
    context.callback = callback;
    context.samples_per_second = frequency;
    context.buffer = malloc(BUFFER_SIZE);
    if (context.buffer == NULL) return E_MEM;

    ad_write_header();
    accel_raw_data_service_subscribe(N_SAMPLES, accel_d
    accel_service_set_sampling_rate((AccelSamplingRate

    return 0;
}
```

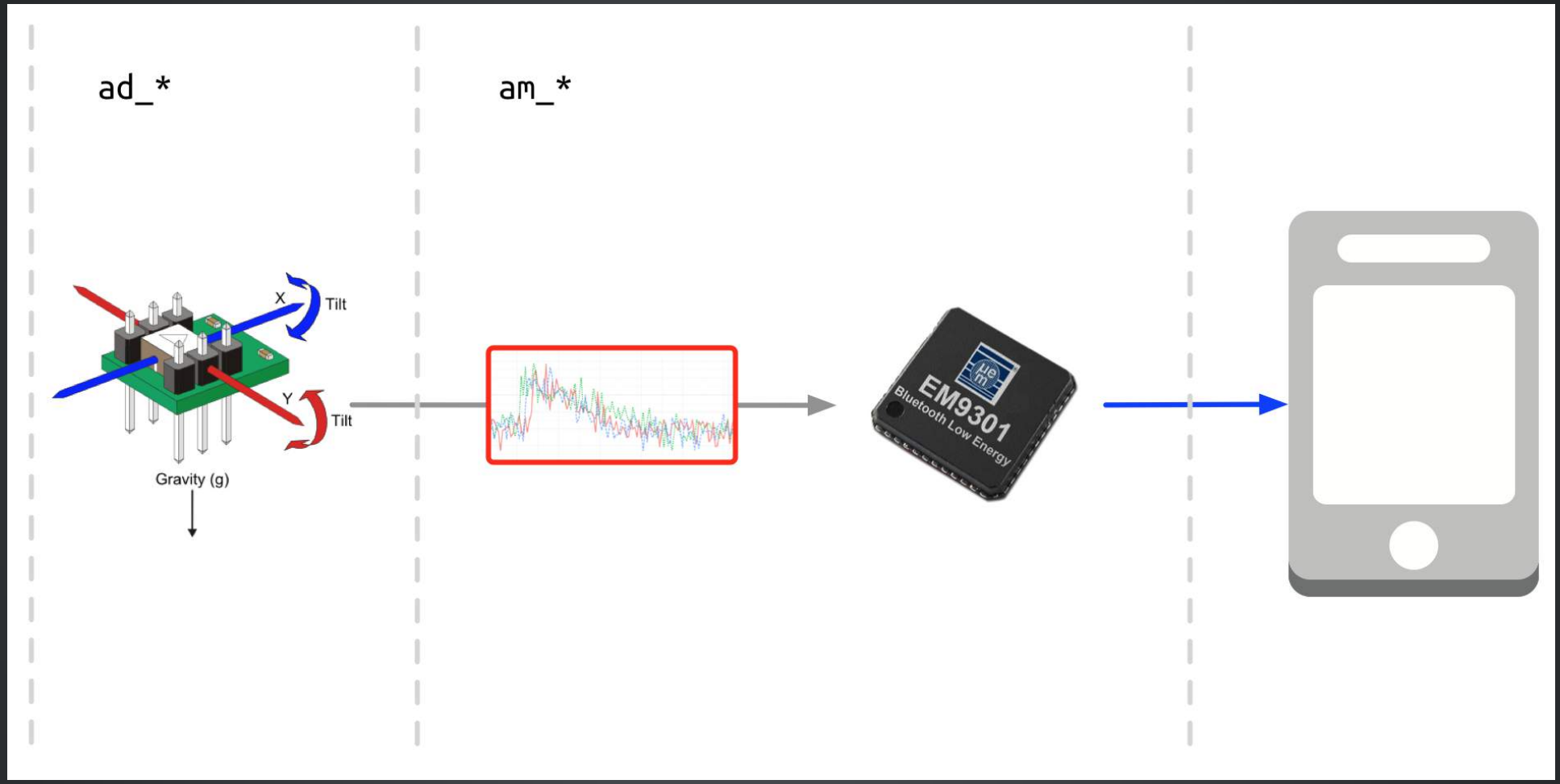
```
#include <gtest/gtest.h>
#include "ad.h"
#include "mock.h"

using namespace pebble::mock;

class ad_test : public testing::Test {
protected:
    static uint8_t *buffer;
    static uint16_t size;
public:
    static void ad_callback(uint8_t *b, uint16_t s);
    virtual ~ad_test();
};
```

```
TEST_F(ad_test, perfectly_aligned) {
    std::vector<AccelRawData> mock_data;
    AccelRawData a = { .x = 1000, .y = 5000, .z = -500
    for (int i = 0; i < 8; ++i) mock_data.push_back(a)

    ad_start(ad_test::ad_callback, 100);
    for (int i = 0; i < 25; ++i) Pebble::accel_service

    ASSERT_TRUE(ad_test::buffer != nullptr);
    auto h = reinterpret_cast<header *>(ad_test::buffe
    EXPECT_EQ(h->type, HEADER_TYPE /* 0xad */);
    EXPECT_EQ(h->count, 100);
    EXPECT_EQ(h->samples_per_second, 100);

    auto data = reinterpret_cast<threed_data *>(
                    ad_test::buffer + sizeof(header));

    for (int i = 0; i < h->count; ++i) { ...}

    ad_stop();
}
```

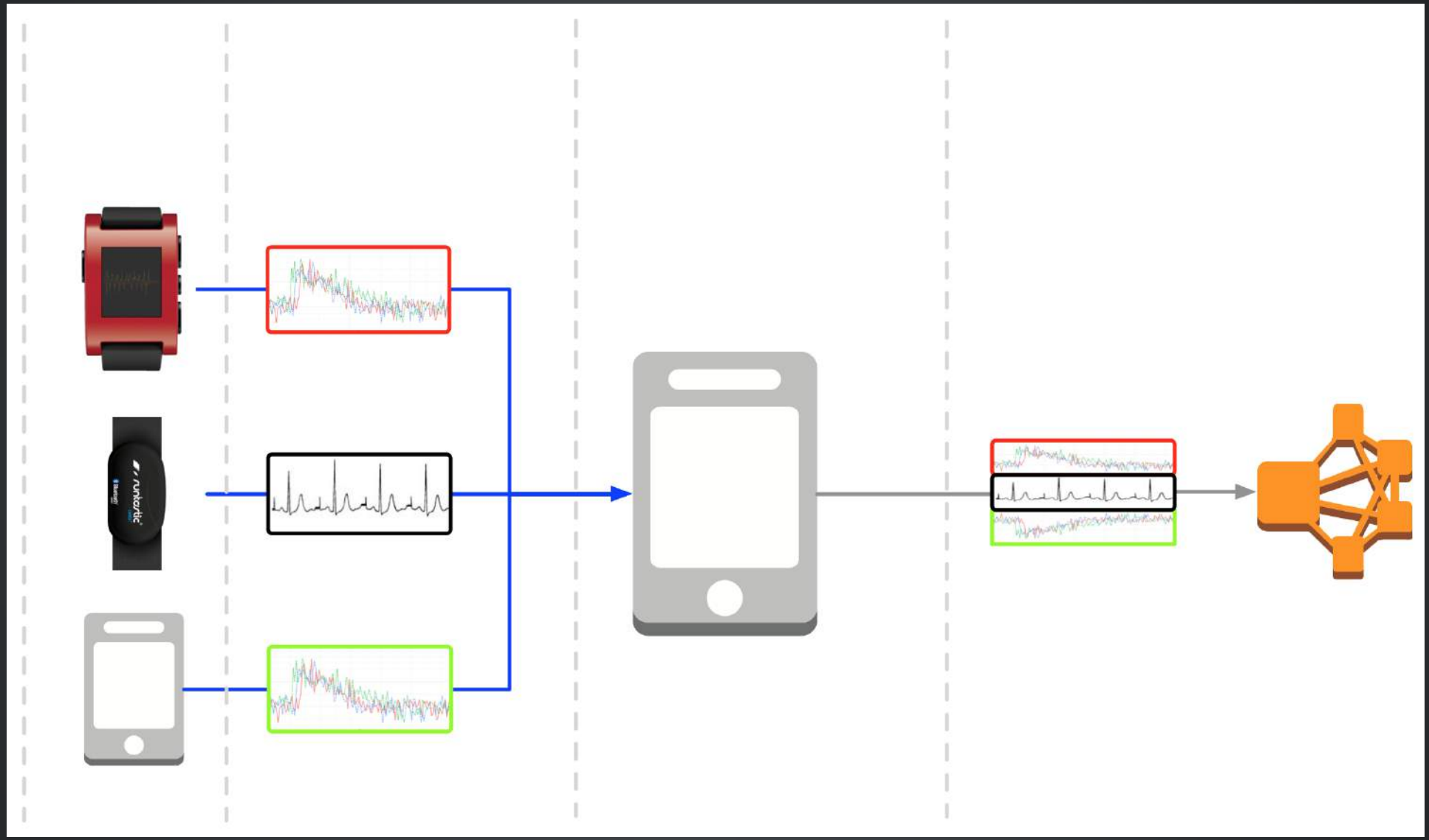
```
#pragma once
#include "gfs.h"
#include "queue.h"

#ifdef __cplusplus
extern "C" {
#endif

    sample_callback_t am_start();
    void am_stop();
    int am_count();
    ...

#ifdef __cplusplus
}
#endif
```

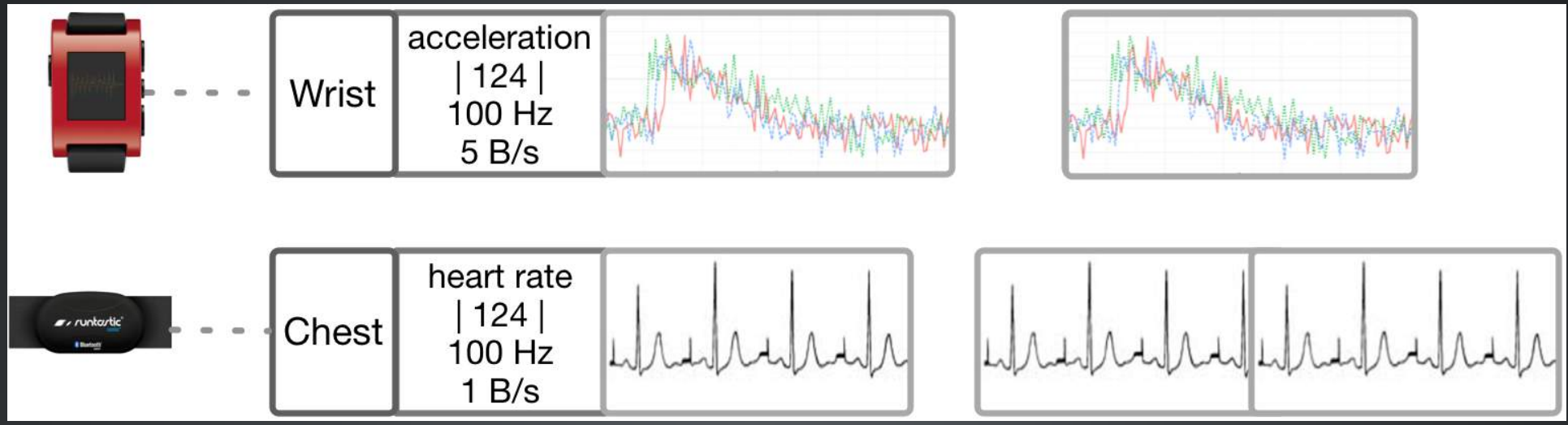
```
void _am_outbox_sent(DictionaryIterator *it, void *ctx) { ... }

void _am_outbox_failed(DictionaryIterator *it, AppMessageResult r, void *c

void _am_sample_callback(uint8_t *buffer, uint16_t size) {
    queue_add(_am_message_queue, buffer, size);
    _send_next_message();
}

void _send_next_message() {
    uint8_t *buffer;
    uint16_t size;
    queue_peek(_am_message_queue, &buffer, &size);
    int queueLength = queue_length(_am_message_queue);

    DictionaryIterator *message;
    app_message_outbox_begin(&message);
    dict_write_data(message, 0xface0fb0, buffer, size);
    dict_write_end(message);
    app_message_outbox_send();
}

sample_callback_t am_start() {
    _am_message_queue = queue_create();

    app_message_open(APP_MESSAGE_INBOX_SIZE_MINIMUM, APP_MESSAGE_OUTBOX_SI
    app_message_register_outbox_sent(_am_outbox_sent);
    app_message_register_outbox_failed(_am_outbox_failed);

    return &_am_sample_callback;
}
```

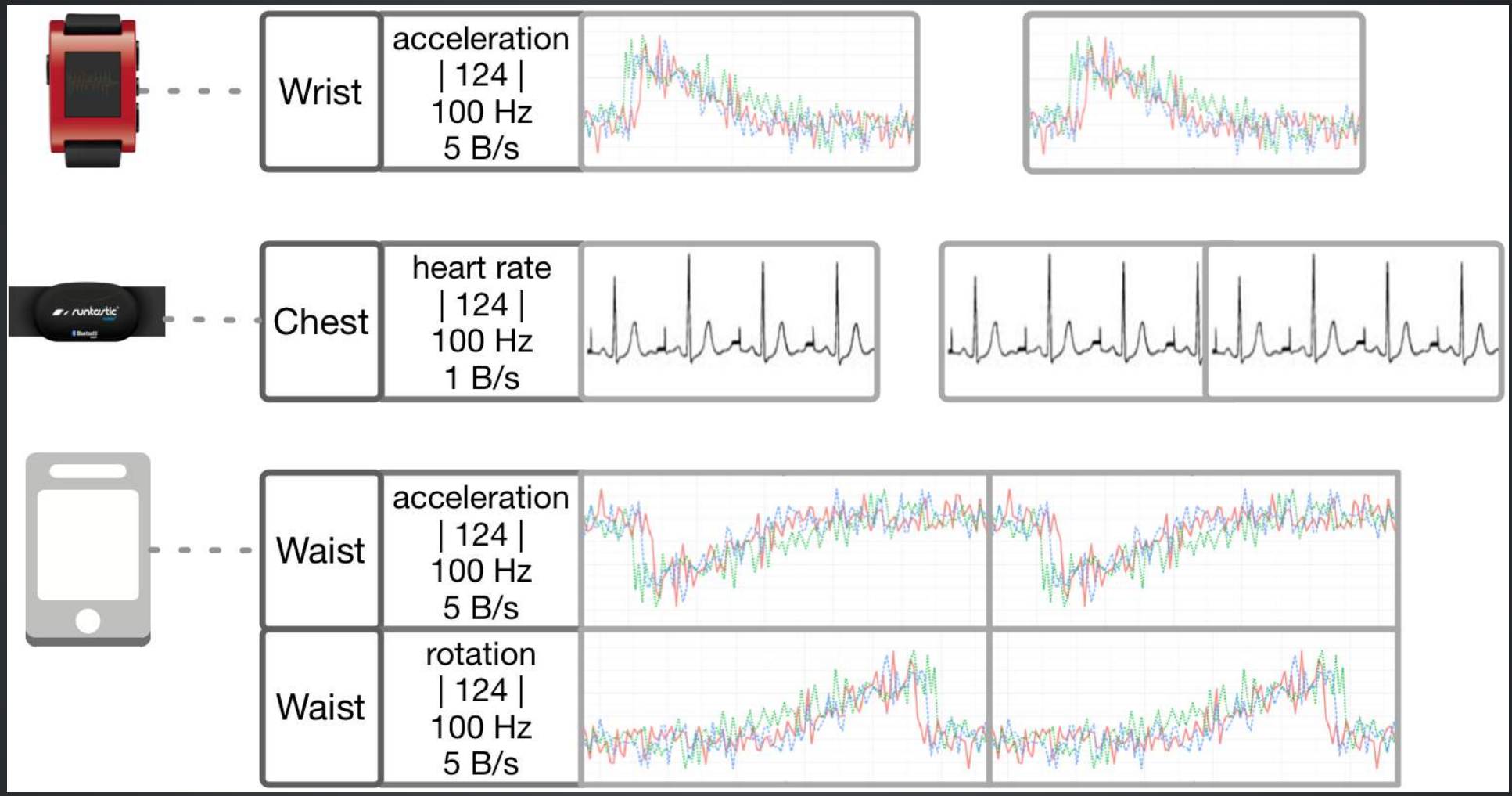
```
static void init(void) {
    window = window_create();
    window_set_window_handlers(window, (WindowHandlers
    window_stack_push(window, true /* Animated */);
    window_set_background_color(window, GColorBlack);

    ad_start(am_start(), 100);
}

static void deinit(void) {
    ad_stop();
    am_stop();

    window_destroy(window);
}

int main(void) {
    init();
    app_event_loop();
    deinit();
}
```

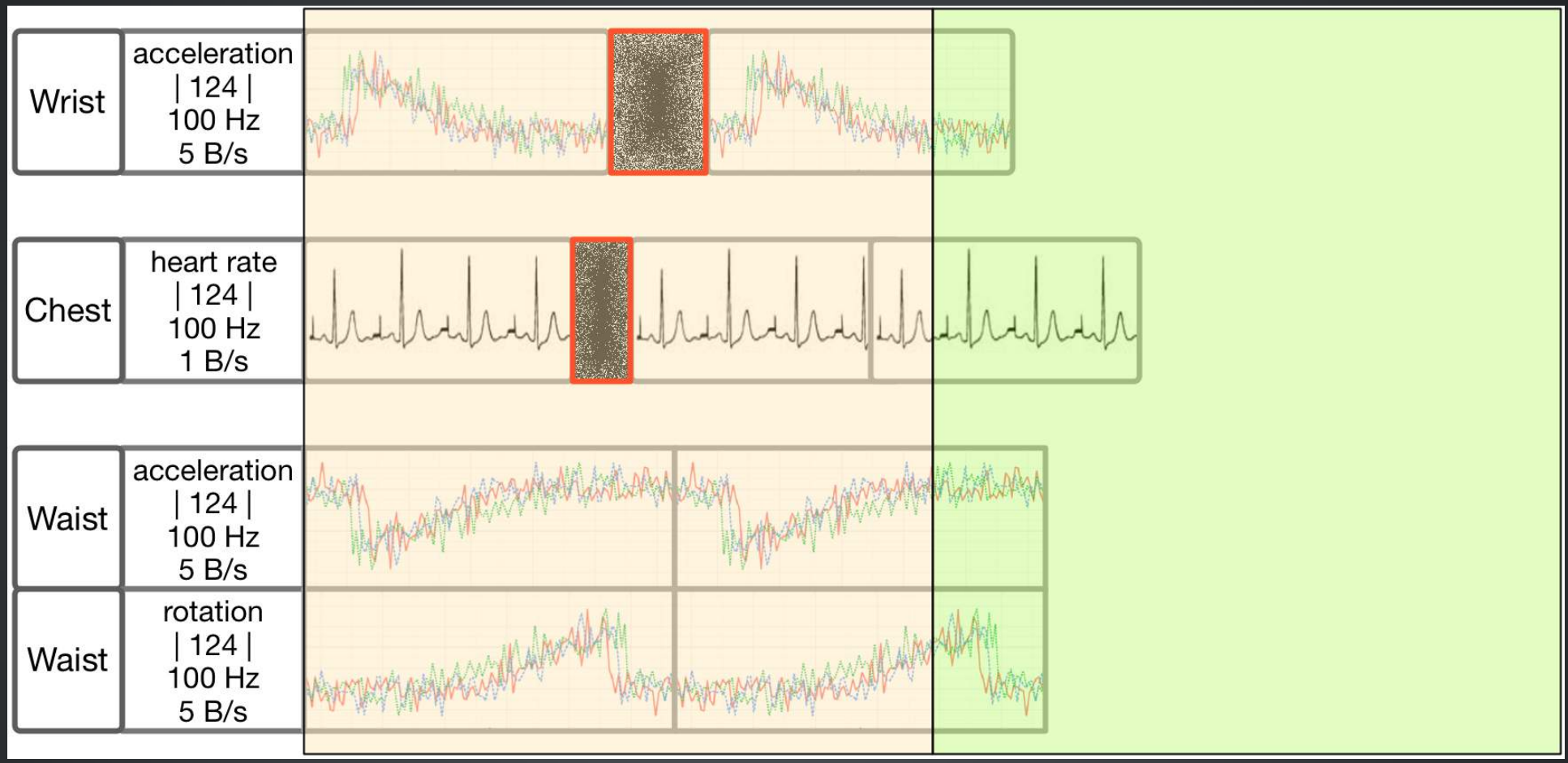

ad_*
am_*
X
Tilt
Y
Tilt
Gravity (g)
EM9301
Bluetooth Low Energy

MOBILE




Wrist
acceleration
| 124 |
100 Hz
5 B/s
Chest
heart rate
| 124 |
100 Hz
1 B/s


Wrist
acceleration
| 124 |
100 Hz
5 B/s
Chest
heart rate
| 124 |
100 Hz
1 B/s
Waist
acceleration
| 124 |
100 Hz
5 B/s
Waist
rotation
| 124 |
100 Hz
5 B/s


Wrist
acceleration
| 124 |
100 Hz
5 B/s
Chest
heart rate
| 124 |
100 Hz
1 B/s
Waist
acceleration
| 124 |
100 Hz
5 B/s
Waist
rotation
| 124 |
100 Hz
5 B/s

SANE MOBILE APPLICATIONS

- Responsive
- Resilient

- CocoaPods
- Swift
- Alamofire
- SwiftZ

```
import Foundation

struct User {
  var id: NSUUID

  struct PublicProfile {
    var firstName: String
    var lastName: String
    var weight: Int?
    var age: Int?

    static func empty() -> PublicProfile {
      return PublicProfile(firstName: "",  lastName: "
                           weight:    nil, age:       ni
    }
  }
}
```

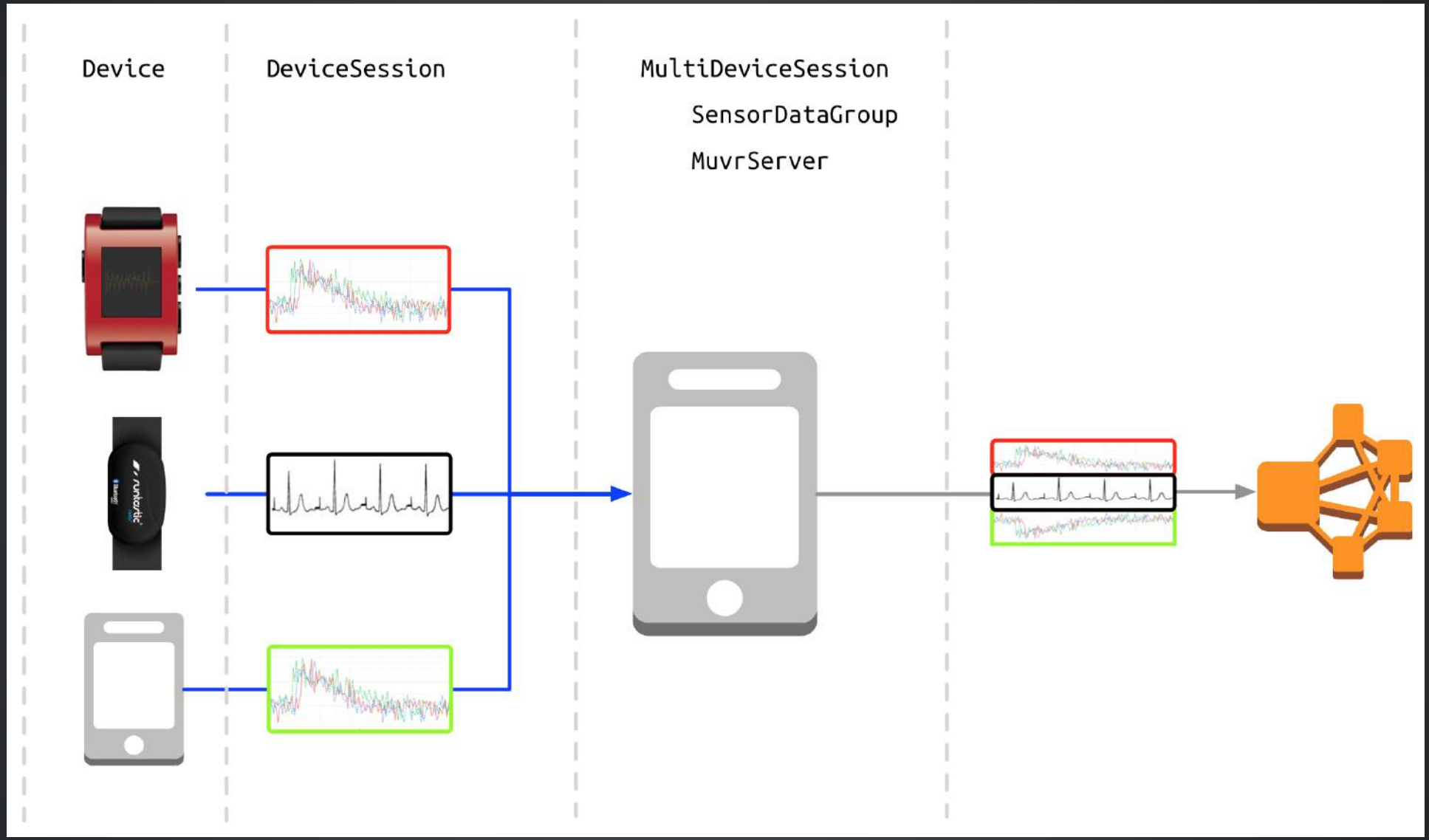
```swift
extension User.PublicProfile {

  static func unmarshal(json: JSON) -> User.PublicProf
    if json.isEmpty { return nil } else {
      return User.PublicProfile(
          firstName: json["firstName"].stringValue,
          lastName:  json["lastName"].stringValue,
          weight:    json["weight"].int,
          age:       json["age"].int)
    }
  }

  func marshal() -> [String : AnyObject] {
    var params: [String : AnyObject] = ["firstName": f
                                        "lastName":  la
    params["age"] = age?
    params["weight"] = weight?
    return params
  }

}
```

```
struct MuvrServerRequest { var path: String; var metho

protocol MuvrServerRequestConvertible {
  var Request: MuvrServerRequest { get }
}

enum MuvrServerURLs : MuvrServerRequestConvertible {
  case UserGetPublicProfile(userId: NSUUID)
  case ExerciseSessionSubmitData(userId: NSUUID, sessi
  ...

  var Request: MuvrServerRequest { get { let r: MuvrSe
    switch self {
      case .UserGetPublicProfile(let userId):
        return MuvrServerRequest(path: "/user/\(userId
                                 method: Method.GET)
      ...
    }()

    return r } } }
}
```

```
public class MuvrServer {

  public class var sharedInstance: MuvrServer {
    struct Singleton { static let instance = MuvrServe
    return Singleton.instance
  }

  private func request(req: MuvrServerRequestConvertib
                       body: Body? = nil) -> Request {
      let msr = req.Request
      // something to the effect of
      return manager.request(msr.method, baseUrlString
  }

  func userGetPublicProfile(userId: NSUUID,
    f: Result<User.PublicProfile?> -> Void) -> Void {
    request(MuvrServerURLs.UserGetPublicProfile(userId
      .completeWith(User.PublicProfile.unmarshal, f)
  }

}
```

```
public enum Result<V> {
  case Error(NSError)
  case Value(Box<V>)

  func cata<U>(l: NSError -> U, r: V -> U) -> U {
    switch self {
      case let Error(e): return l(e)
      case let Value(v): return r(v.value)
    }
  }

  func flatMap<U>(f: V -> Result<U>) -> Result<U> { ..

  static func error(e: NSError) -> Result<V> { ... }

  static func value(v: V) -> Result<V> { ... }
}
```

```
class HomeController : UIViewController, ... {

  private func showProfile(profile: User.PublicProfile
  private func showError(error: NSError) { ... }

  override func viewDidAppear(animated: Bool) {
    MuvrServer
      .sharedInstance
      .userGetPublicProfile(CurrentUser.userId!) {
        $0.cata(showError, showProfile);
      }
  }

}
```


Device
DeviceSession
MultiDeviceSession
SensorDataGroup
MuvrServer

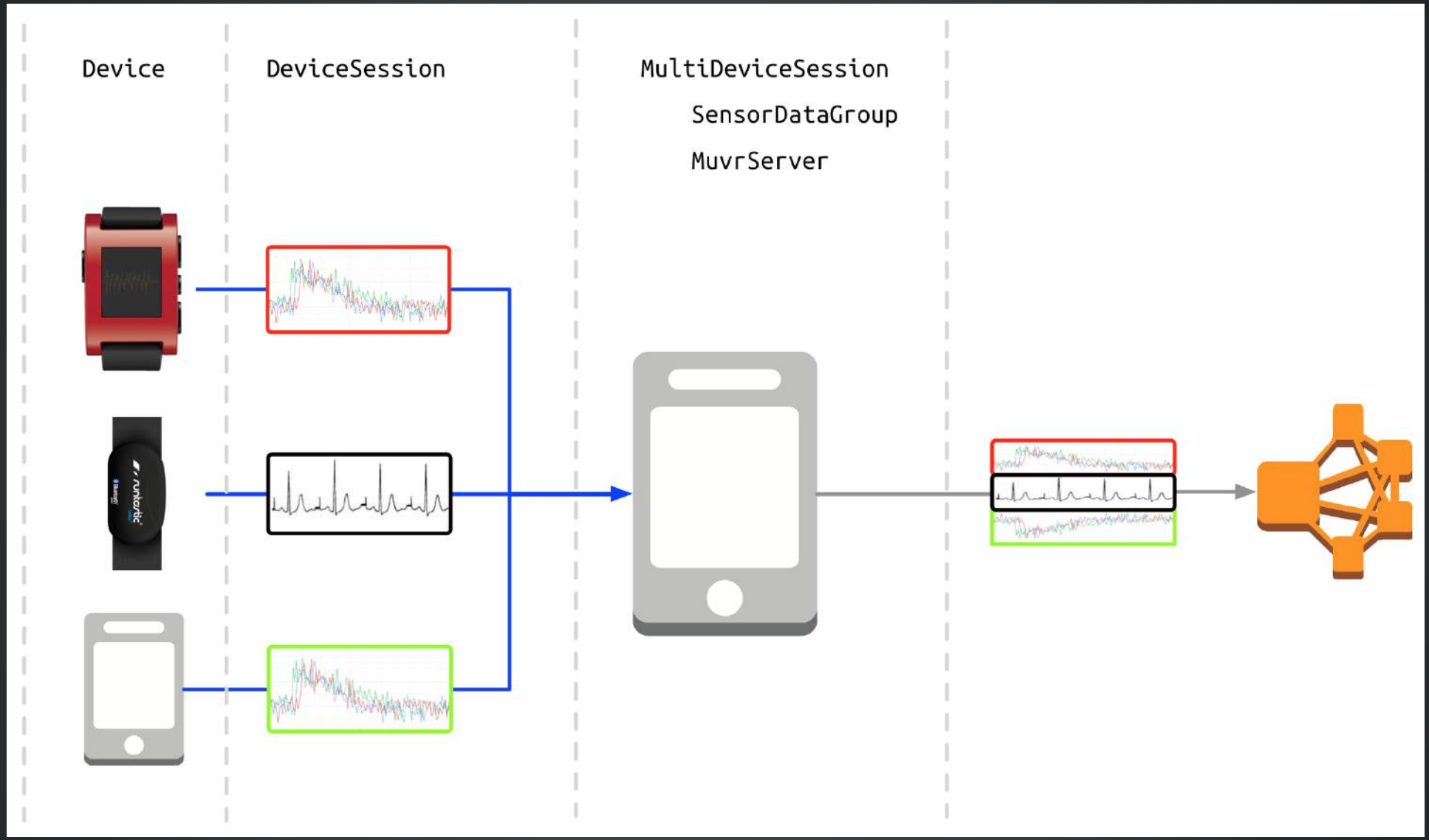
```
class PebbleDeviceSession : DeviceSession {
  private var updateHandler: AnyObject?
  private let adKey   = NSNumber(uint32: 0xface0fb0)
  private let deadKey = NSNumber(uint32: 0x0000dead)

  required init(deviceId: NSUUID, watch: PBWatch!) {
    super.init()
    self.updateHandler = watch.appMessagesAddReceiveUpd
      appMessagesReceiveUpdateHandler)
  }

  private func appMessagesReceiveUpdateHandler(watch: P
    data: [NSObject : AnyObject]!) -> Bool {
    if let x = data[adKey] as? NSData {
      accelerometerDataReceived(x)
    } else if data[deadKey] != nil {
      stop()
    }
    return true
  }
}
```

```c
#include <stdint.h>

typedef struct __attribute__((__packed__)) {
    int16_t x_val : 13;
    int16_t y_val : 13;
    int16_t z_val : 13;
    uint8_t valid : 1;
} threed_data;

void encode_threed_data(int16_t x, int16_t y, int16_t z

void decode_threed_data(const void *buffer,
                        int16_t *x, int16_t *y, int16_t
```

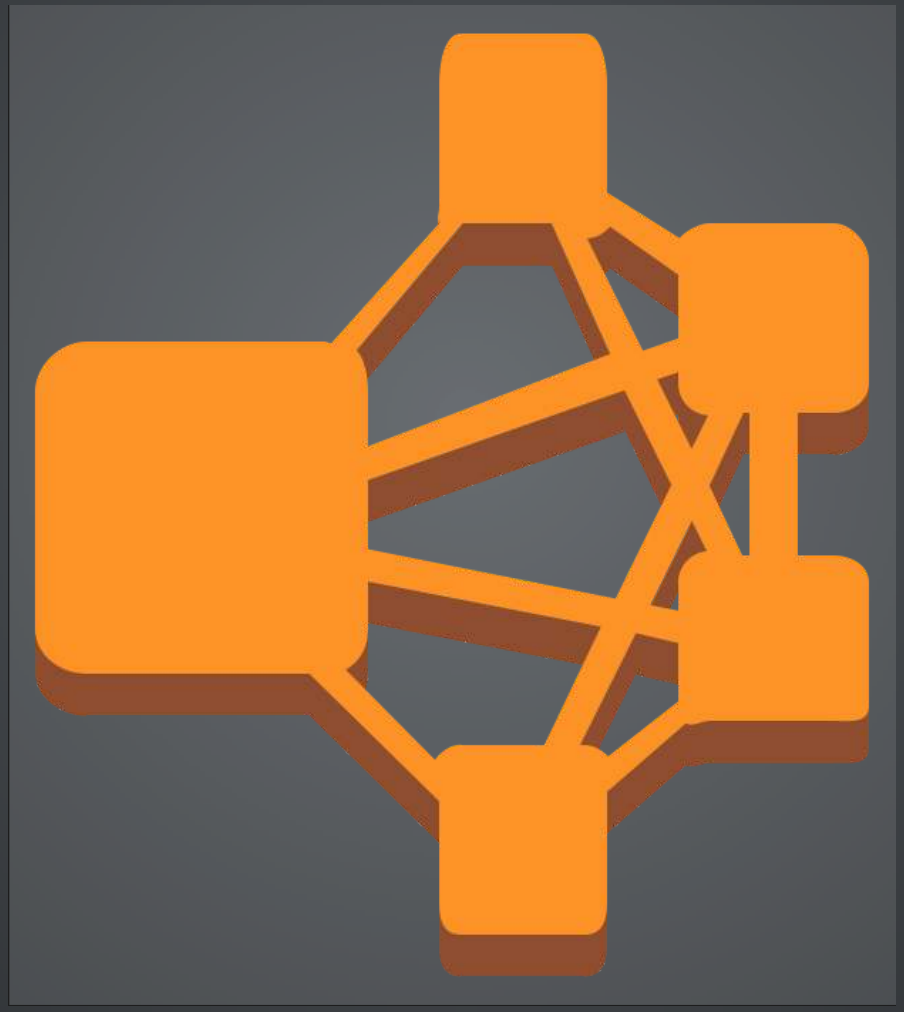
```c
#include "SensorDataProtocol.h"

#define SIGNED_12_MAX(x) \
  (int16_t)((x) > 4095 ? 4095 : ((x) < -4095 ? -4095 :

void encode_threed_data(int16_t x, int16_t y, int16_t :
    threed_data *tdd = (threed_data *)data;
    tdd->x_val = SIGNED_12_MAX(x);
    tdd->y_val = SIGNED_12_MAX(y);
    tdd->z_val = SIGNED_12_MAX(z);
}

void decode_threed_data(const void *buffer,
                        int16_t *x, int16_t *y, int16_t
    threed_data *data = (threed_data *)buffer;
    (*x) = data->x_val;
    (*y) = data->y_val;
    (*z) = data->z_val;
}
```

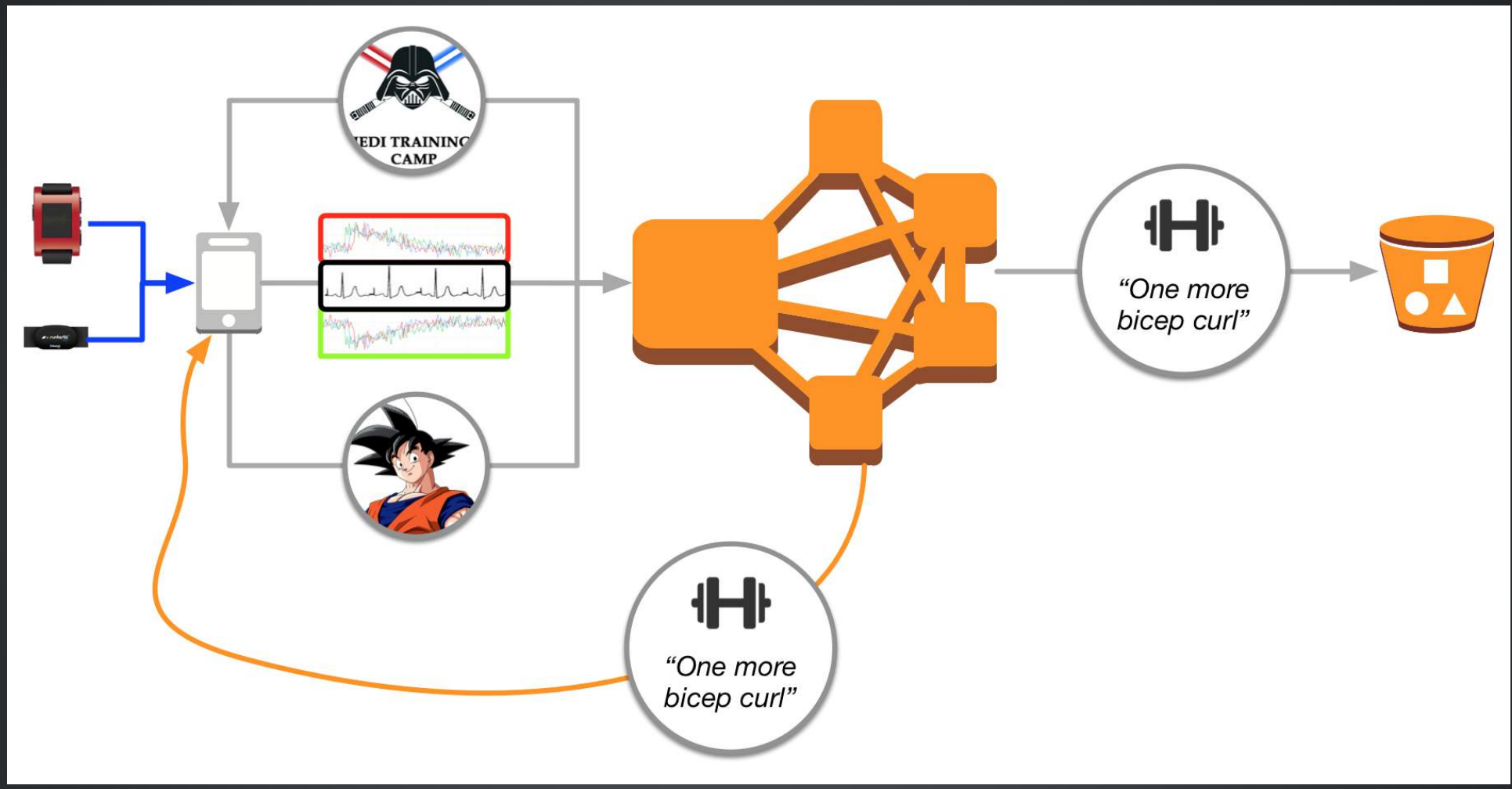
```
class MultiDeviceSession : DeviceSession,
  DeviceSessionDelegate, DeviceDelegate, SensorDataGroupBufferDelegate {
  private let combinedStats = DeviceSessionStats<DeviceSessionStatsTypes.K
  private let multiDeviceId = DeviceId(UUIDString: "00000000-0000-0000-000
  private let sensorDataGroupBuffer: SensorDataGroupBuffer!
  private var devices: [ConnectedDevice] = []

  required init(...) {
    super.init()
    self.sensorDataGroupBuffer = SensorDataGroupBuffer(...)
    for device in Devices.devices {
      device.connect(onDone: { d in self.devices += [d] })
    }
  }

  func start() -> Void { for d in devices { d.start() } }

  func deviceSession(session: DeviceSession, sensorDataReceivedFrom device
    atDeviceTime time: CFAbsoluteTime, data: NSData) {
    sensorDataGroupBuffer
      .decodeAndAdd(data, fromDeviceId: deviceId, maximumGap: 0.3, gapValu

    combinedStats.merge(session.getStats()) { k in
        let location = ...
        return DeviceSessionStatsTypes.KeyWithLocation(
          sensorKind: k.sensorKind, deviceId: k.deviceId, location: locati
    }
  }
}
```

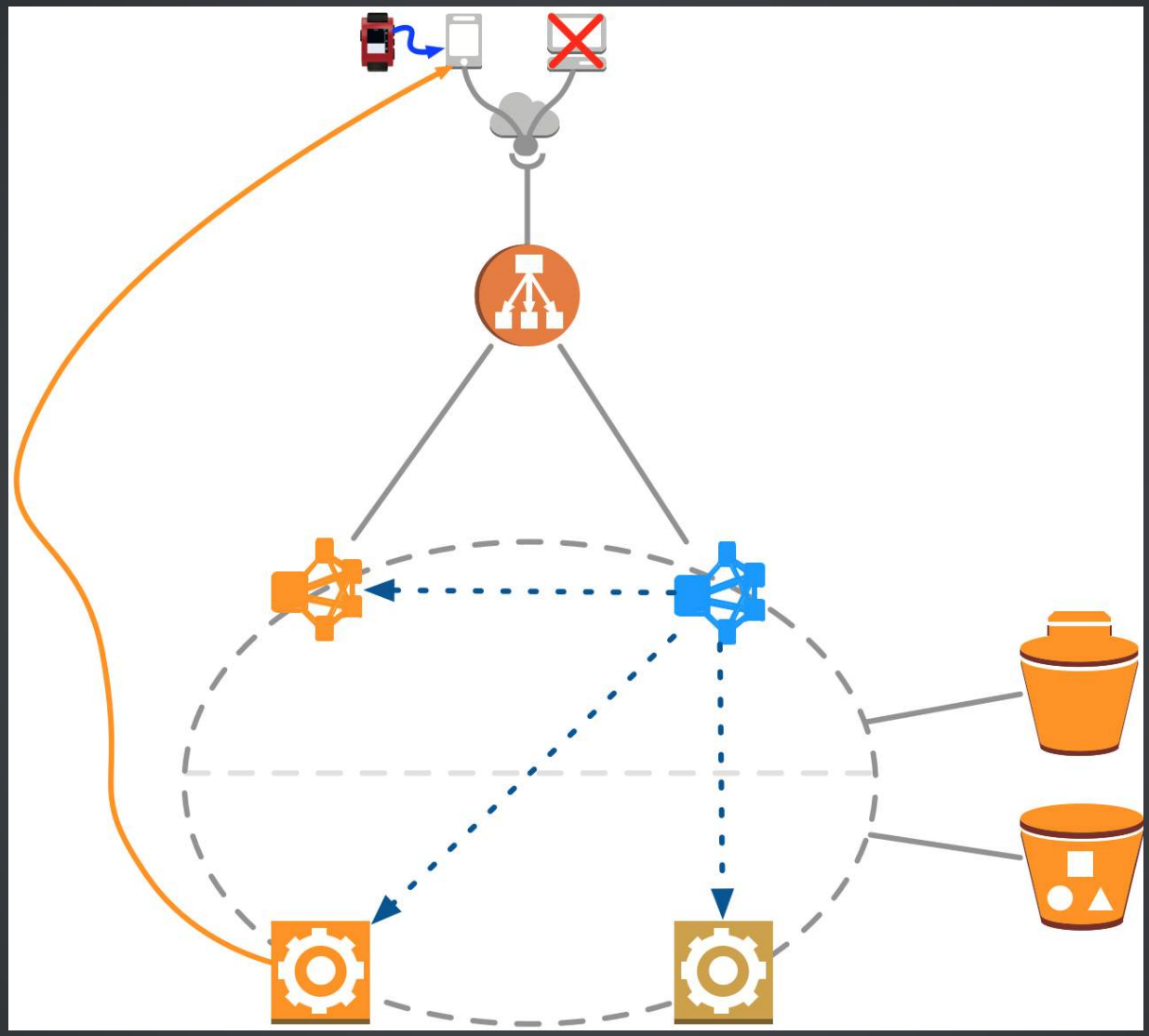
```
class LiveSessionController: UIPageViewController,
  ExerciseSessionSettable, DeviceSessionDelegate,
  DeviceDelegate, MultiDeviceSessionDelegate {
  private var multi: MultiDeviceSession?

  func setExerciseSession(session: ExerciseSession) {
    self.exerciseSession = session
    multi = MultiDeviceSession(delegate: self,
                deviceDelegate: self, deviceSessionDeleg
    multi!.start()
    UIApplication.sharedApplication().idleTimerDisable
  }

  func deviceSession(session: DeviceSession,
    sensorDataReceivedFrom deviceId: DeviceId,
    atDeviceTime: CFAbsoluteTime, data: NSData) {
    if let x = exerciseSession {
      x.submitData(data, f: const(()))
    }
  }
}
```

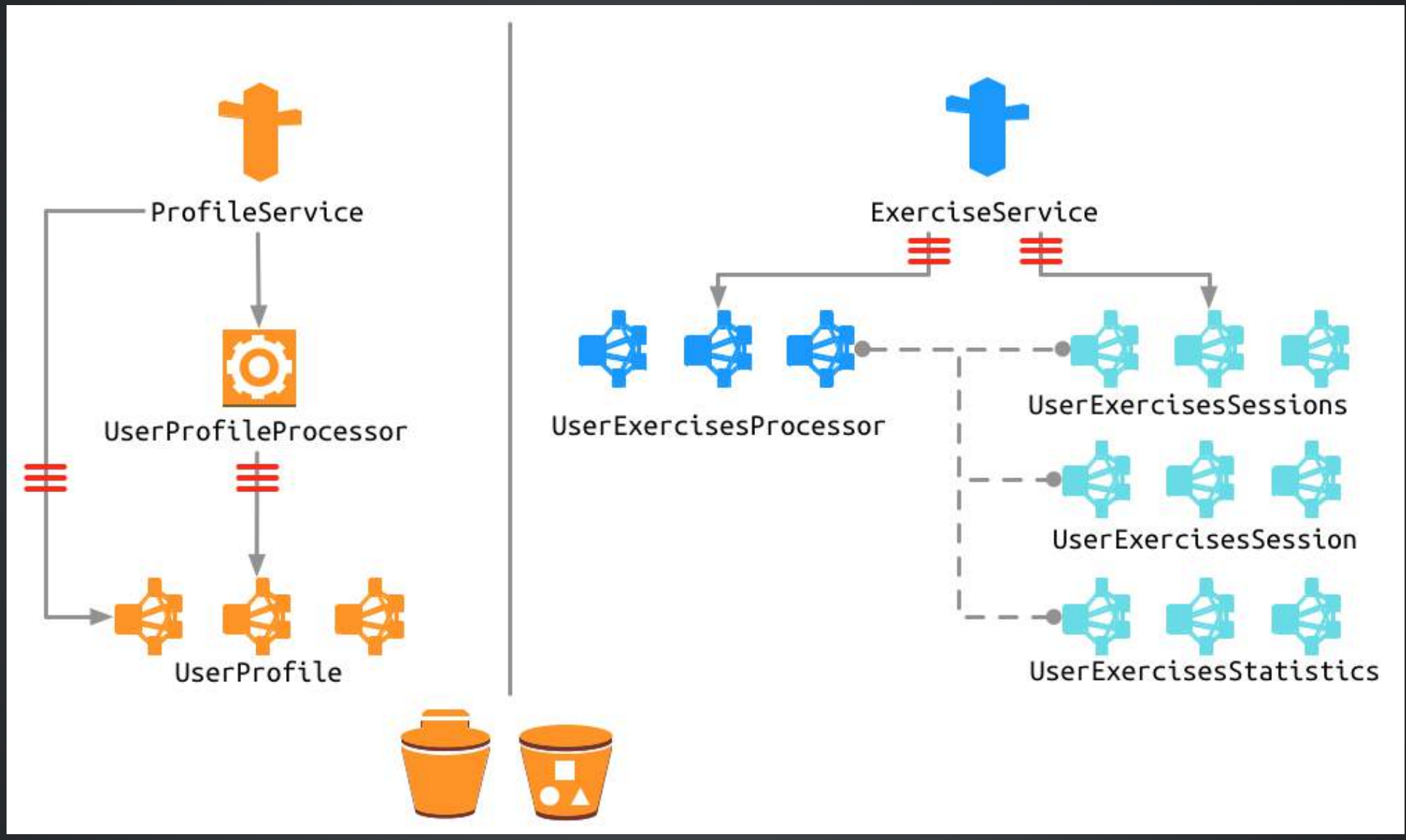

Device
DeviceSession
MultiDeviceSession
SensorDataGroup
MuvrServer

AKKA




EDI TRAINING
CAMP
"One more
bicep curl"
"One more
bicep curl"




ProfileService
UserProfileProcessor
UserProfile
ExerciseService
UserExercisesProcessor
UserExercisesSessions
UserExercisesSession
UserExercisesStatistics

```
trait ProfileService extends Directives
  with CommonMarshallers with CommonPathDir

  def userProfileRoute(userProfile: ActorRe
                       userProfileProcessor
                      (implicit _: Executio
    path("user") ~
    path("user" / UserIdValue) ~
    path("user" / UserIdValue / "check") ~
    path("user" / UserIdValue / "image") ~
    path("user" / UserIdValue / "device" /
    path("user" / UserIdValue / "device" /

}
```

```
class UserProfileProcessor(userProfile: Act
  extends PersistentActor with ActorLogging

  override def receiveRecover: Receive = ..

  override def receiveCommand: Receive = ..

}
```

```
var accounts: KnownAccounts = KnownAccounts

override def receiveCommand: Receive = {
  case UserRegister(email, _)
  if accounts.contains(email) ⇒
    sender() ! \/.left("Username already ta
}
```

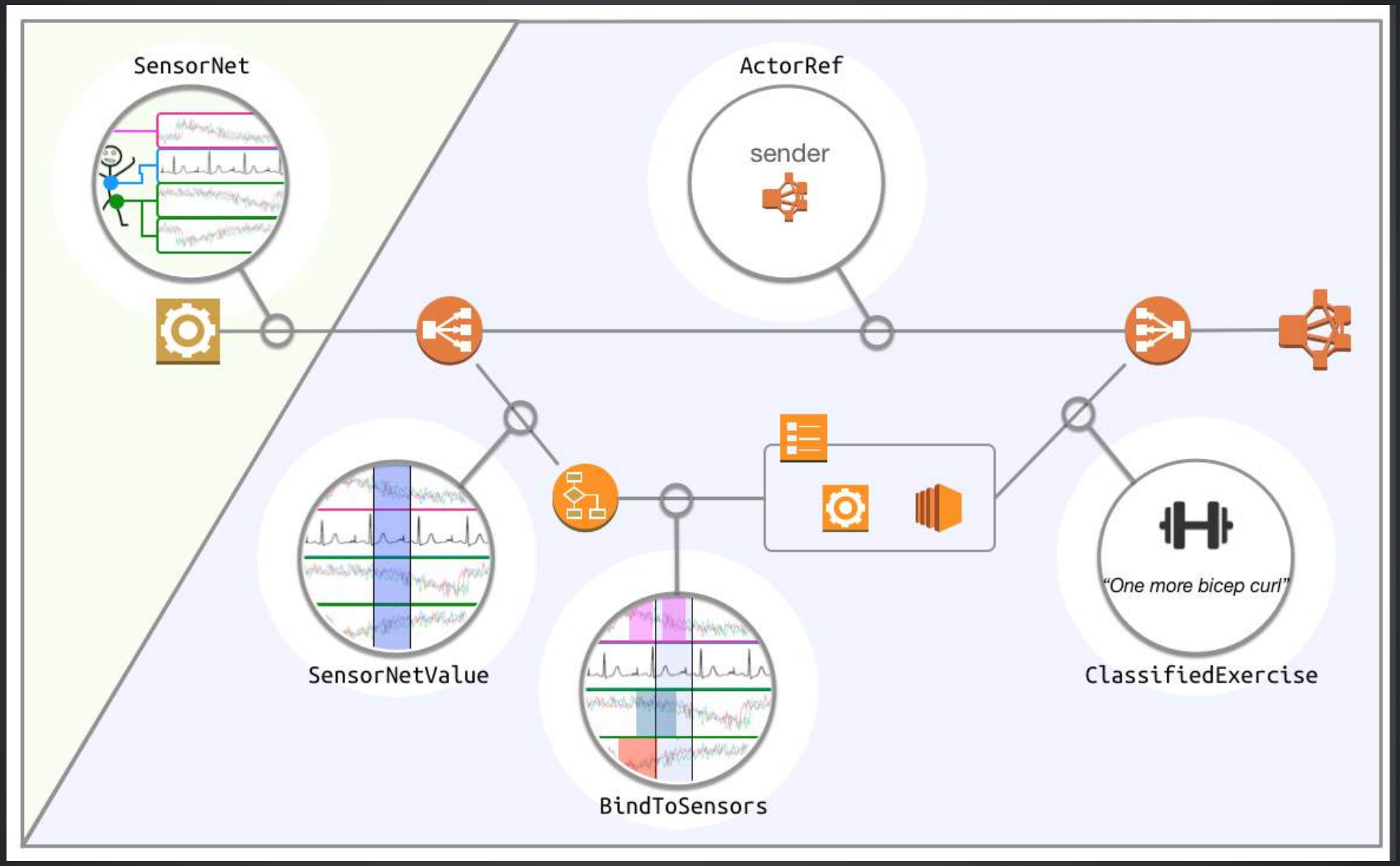
```
var accounts: KnownAccounts = KnownAccounts

override def receiveCommand: Receive = {
  ...
  case UserRegister(email, password) ⇒
    persist(UserRegistered(UserId.randomId(
      userRegistered ⇒
      userProfile ! userRegistered
      accounts = accounts.withNewAccount(..
      saveSnapshot(accounts)

      sender() ! \/.right(userRegistered.us
    }
}
```

```
class UserProfile extends PersistentActor {
  private var profile: Profile = _
  override val persistenceId: String =
    s"user-profile-${self.path.name}"

  override def receiveCommand: Receive = no

  private def notRegistered: Receive = ...

  private def registered: Receive = ...

}
```

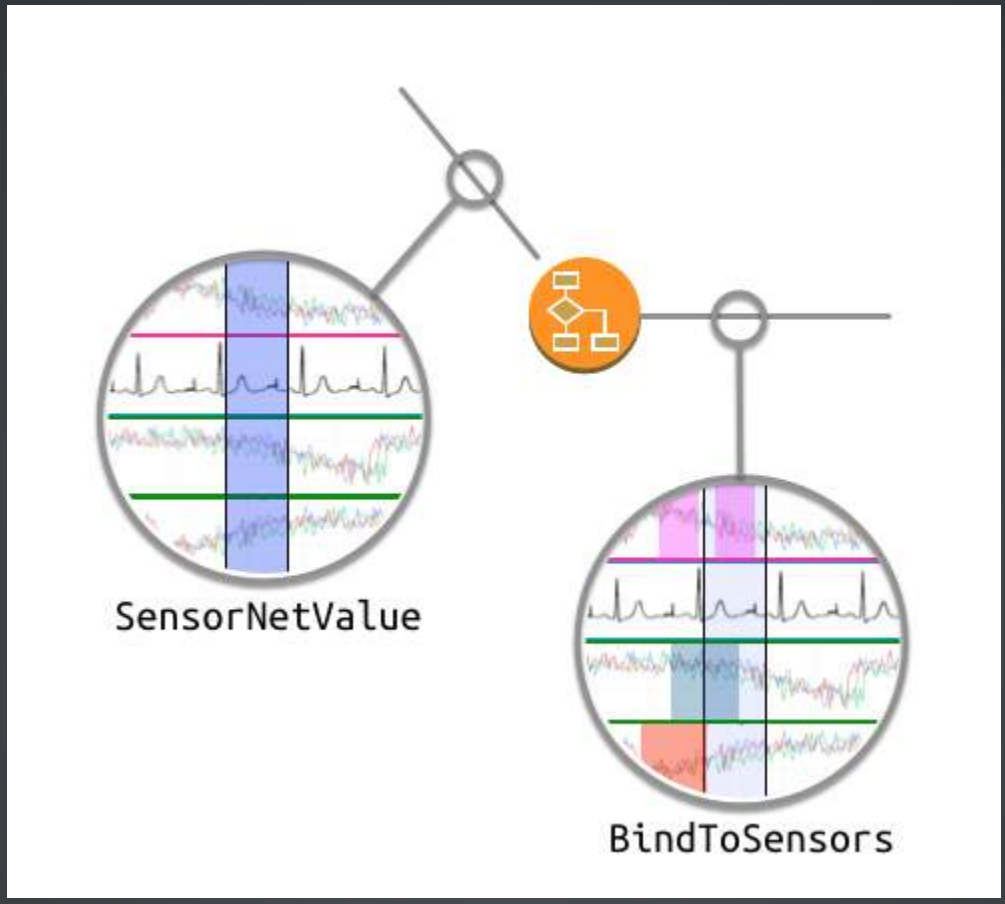
```
private var profile: Profile = _

private def notRegistered: Receive = {
  case cmd: Account ⇒
    persist(cmd) { acc ⇒
      profile = Profile(acc, Devices.empty,
      saveSnapshot(profile)
      context.become(registered)
    }
}
private def registered: Receive = {
  case GetAccount ⇒ sender() ! profile.acc
  ...
}
```

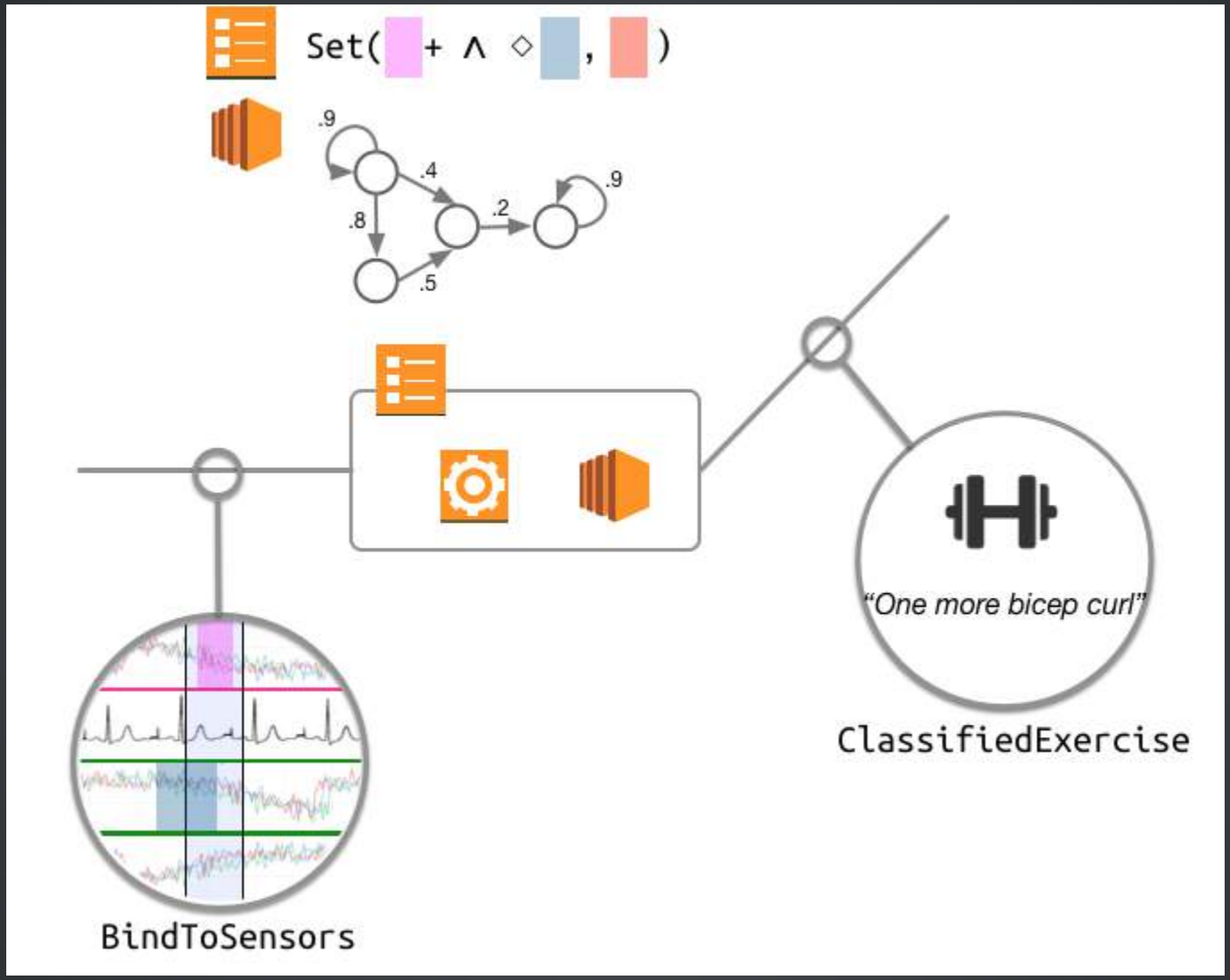
```
class UserExercisesProcessor(notification:
  userProfile: ActorRef) extends Persistent

  override def receiveRecover: Receive = ..
  override def receiveCommand: Receive = no

  def replaying(oldId: SessionId, newId: Se
        sessionProps: SessionProperties): R

  def exercising(id: SessionId,
        sessionProps: SessionProperties): R

  def notExercising: Receive = ...
}
```

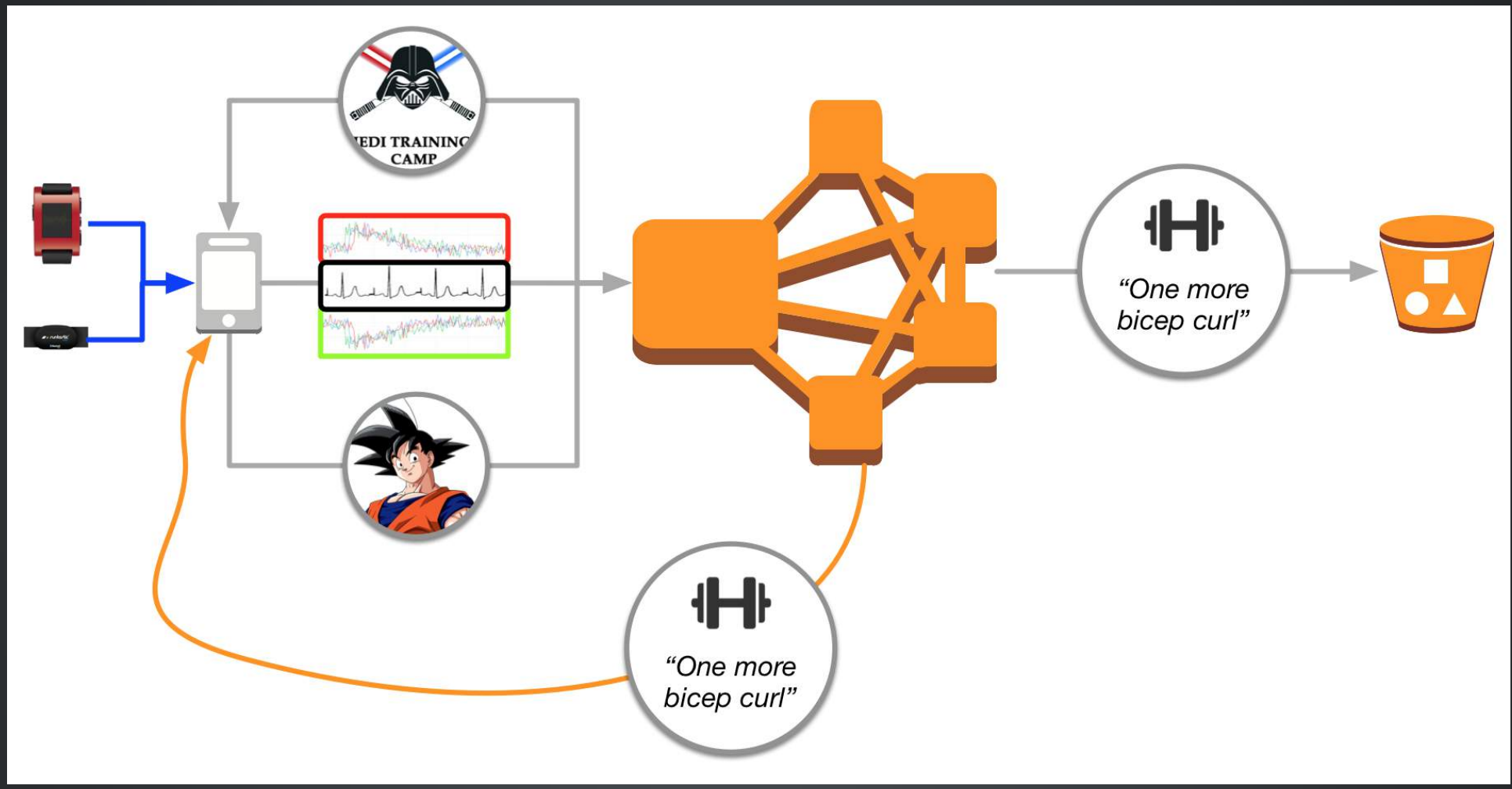
```
def exercising(id: SessionId,
  sessionProps: SessionProperties): Receive
  case ExerciseDataProcessMultiPacket(`id`,
    classifier ! Classify(packet)

  case ClassifiedExercise(md, _, exercise)
    persist(ExerciseEvt(id, md, exercise))(

  case NoMovement ⇒
    persist(NoMovement(id))(unit)

  case NoExercise(metadata) ⇒
    persist(NoExerciseEvt(id, metadata))(un
}
```


SensorNet
ActorRef
sender
SensorNetValue
BindToSensors
"One more bicep curl"
ClassifiedExercise

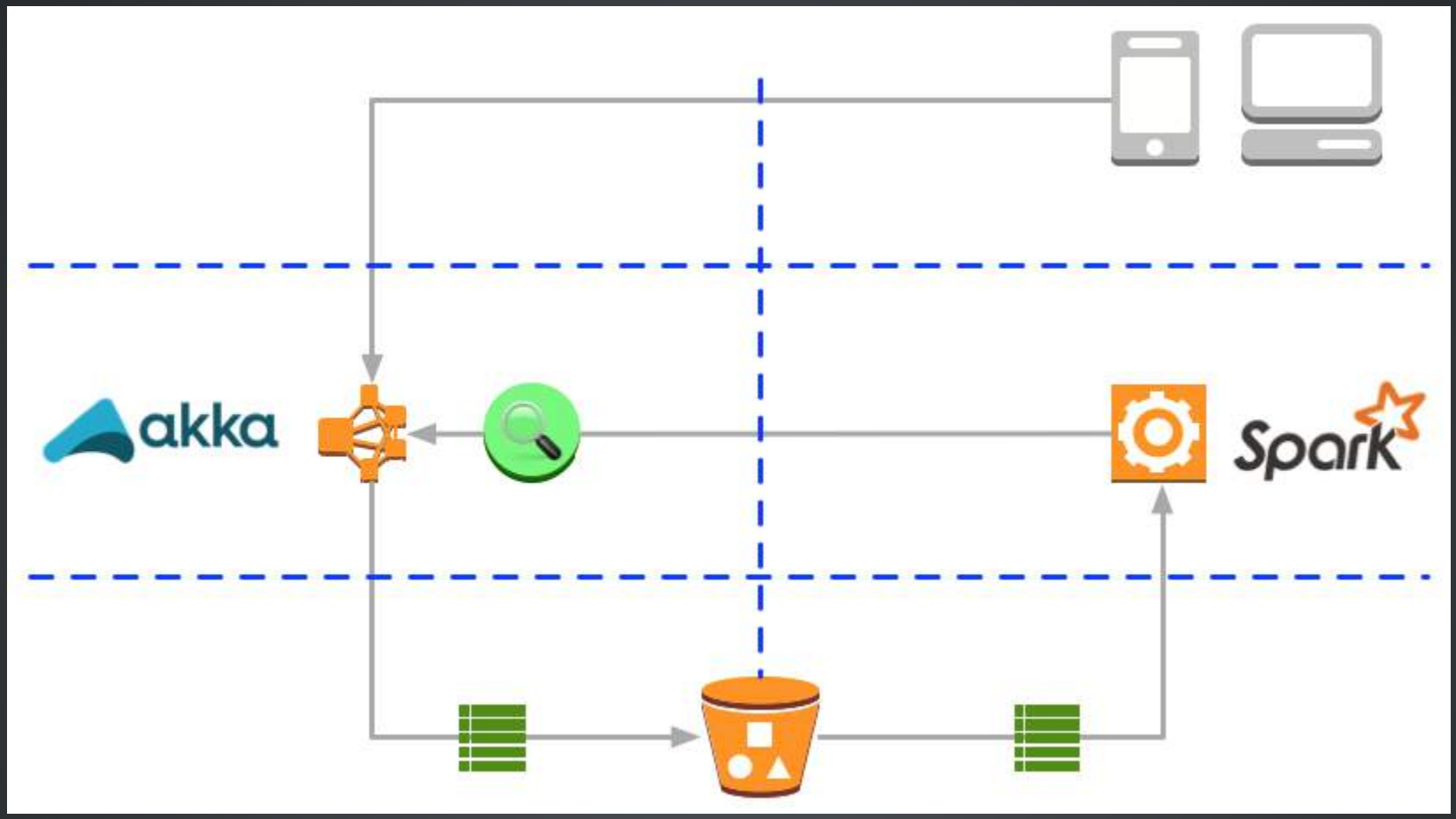

ActorRef

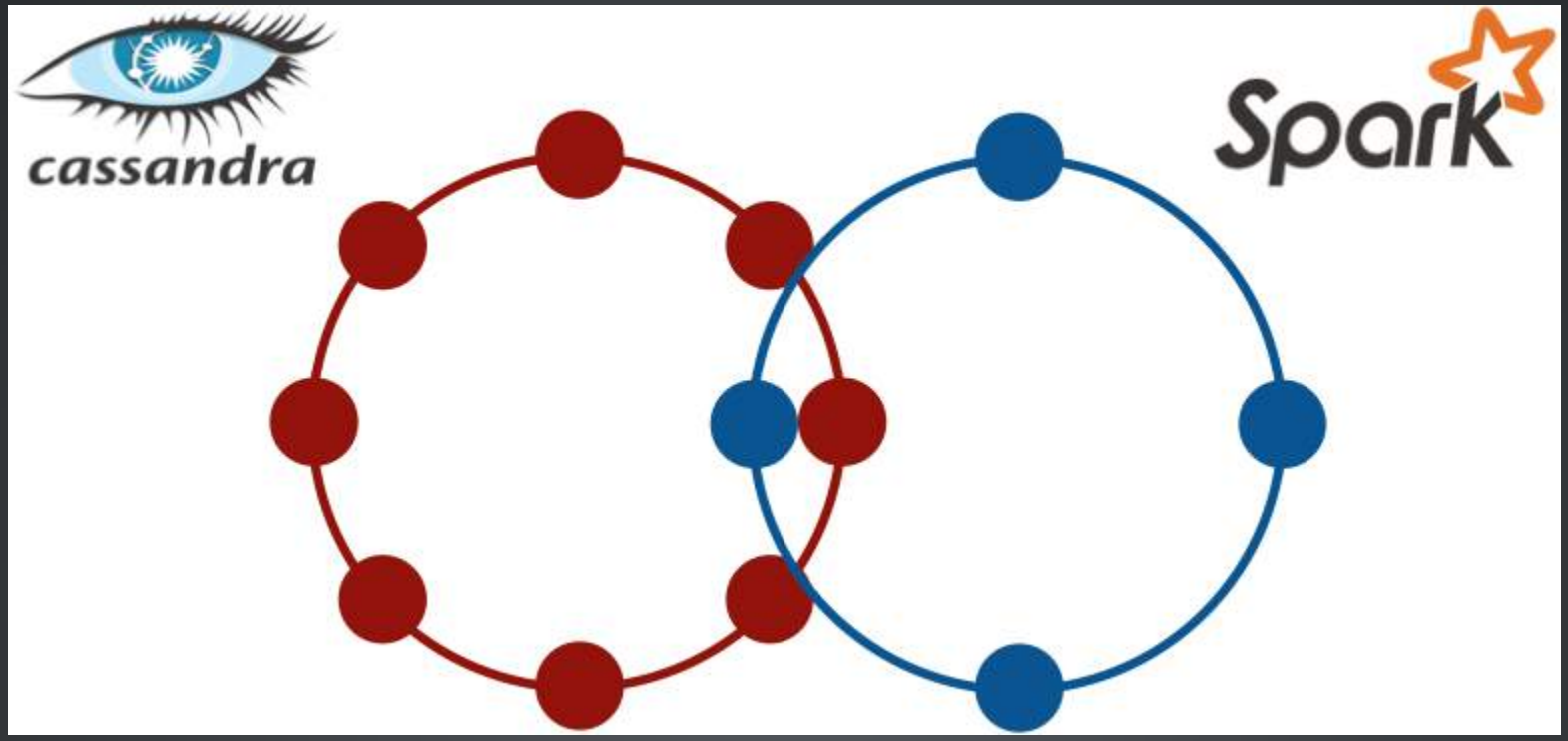

SensorNetValue
BindToSensors


Set(+ ∧ ◇ ,)
.9
.4
.8
.2
.9
.5
"One more bicep curl"
ClassifiedExercise
BindToSensors

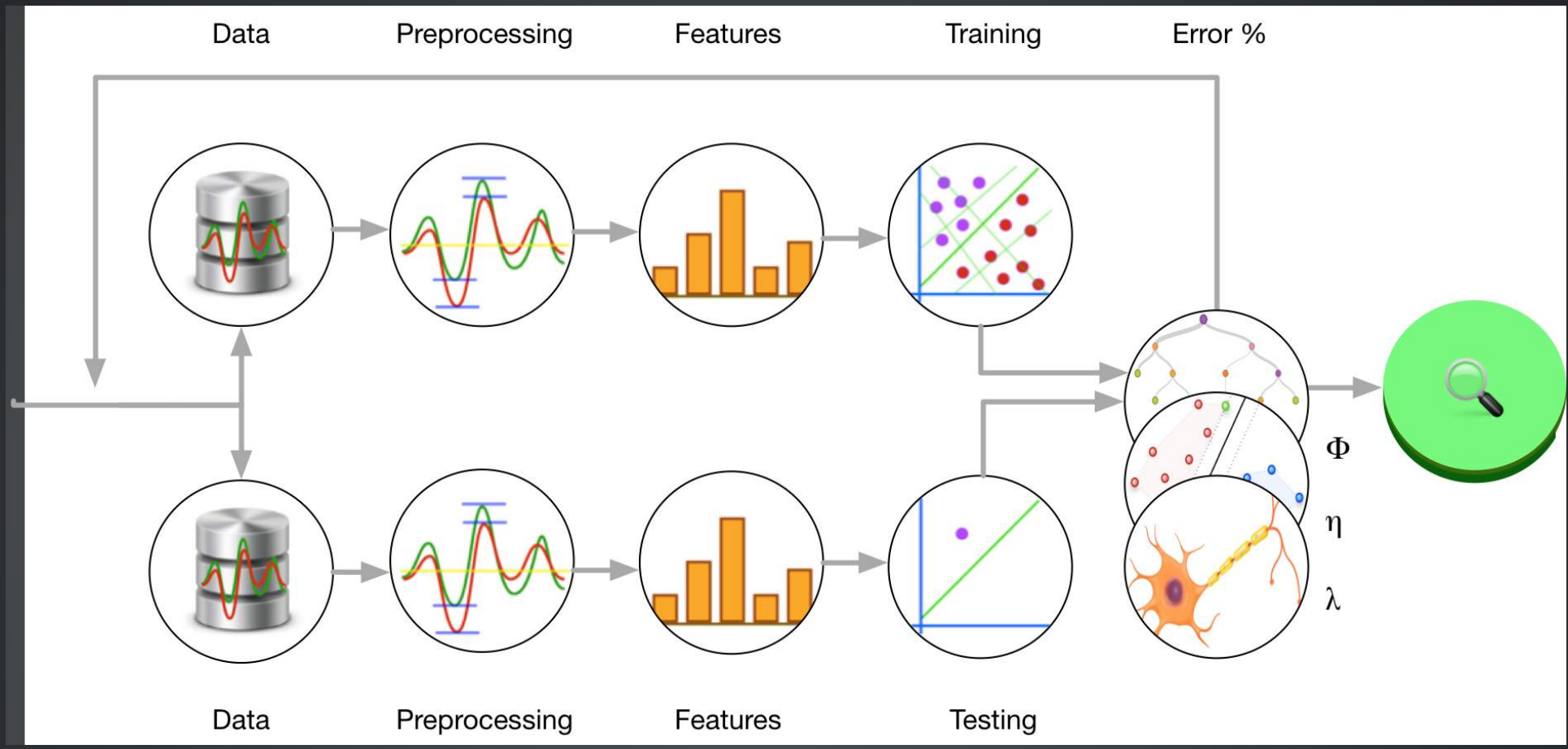

EDI TRAINING
CAMP
"One more bicep curl"
"One more bicep curl"

APACHE SPARK


akka
Spark

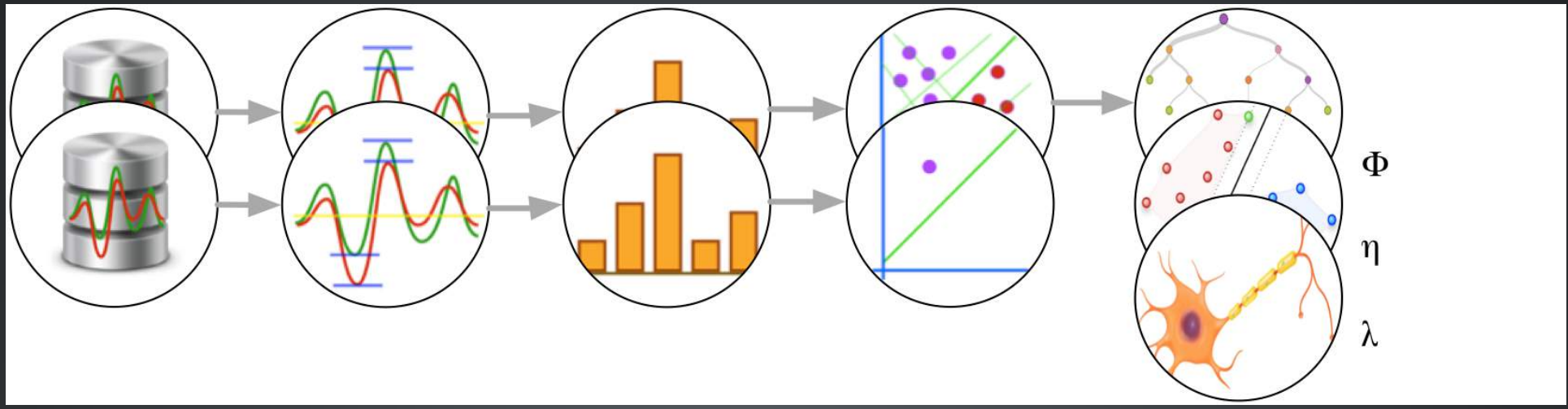

cassandra
Spark

```
class JobManager(
    override val master: String,
    override val config: Config)
  extends Actor
  with Driver
  with ActorLogging
  with PipeToSupport {

  override def receive: Receive = {
    case BatchJobSubmit('Suggestions) ⇒
      submit(Job[SuggestionsJob], ()).pipeT
  }
}
```

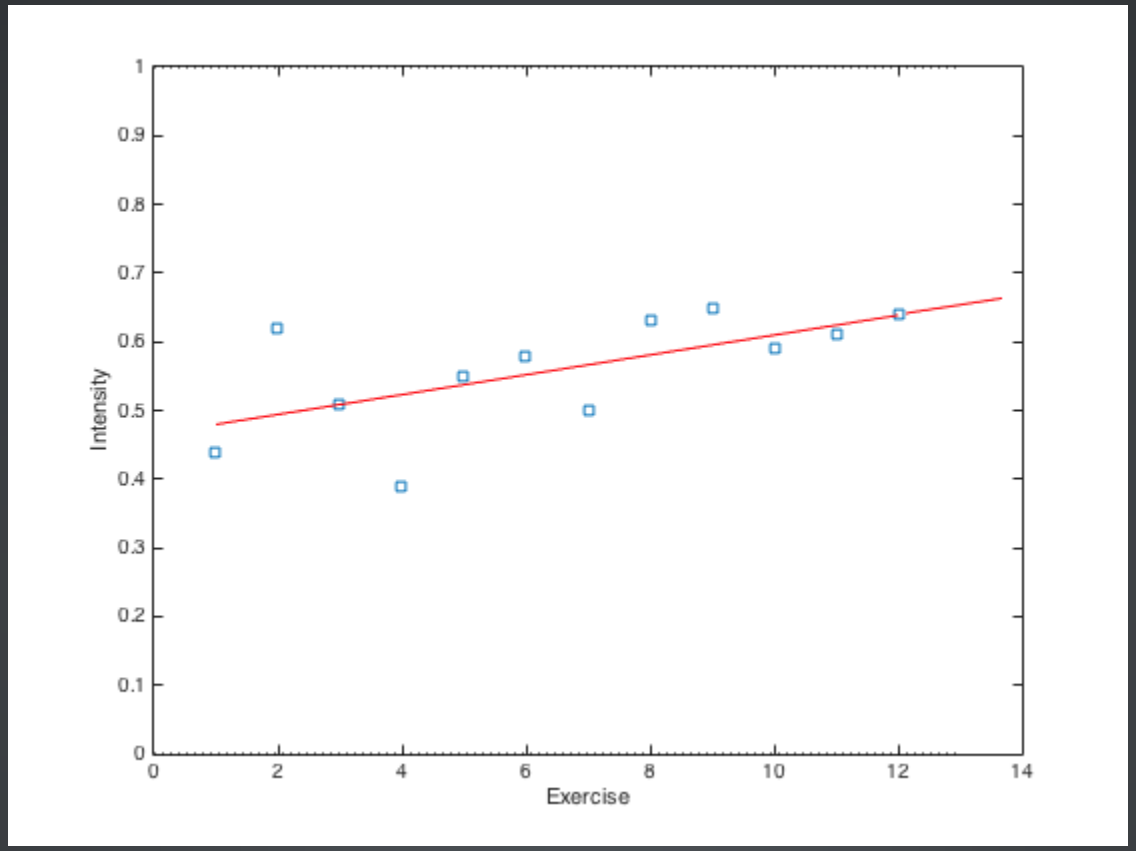

Data
Preprocessing
Features
Training
Error %
Φ
η
λ
Data
Preprocessing
Features
Testing

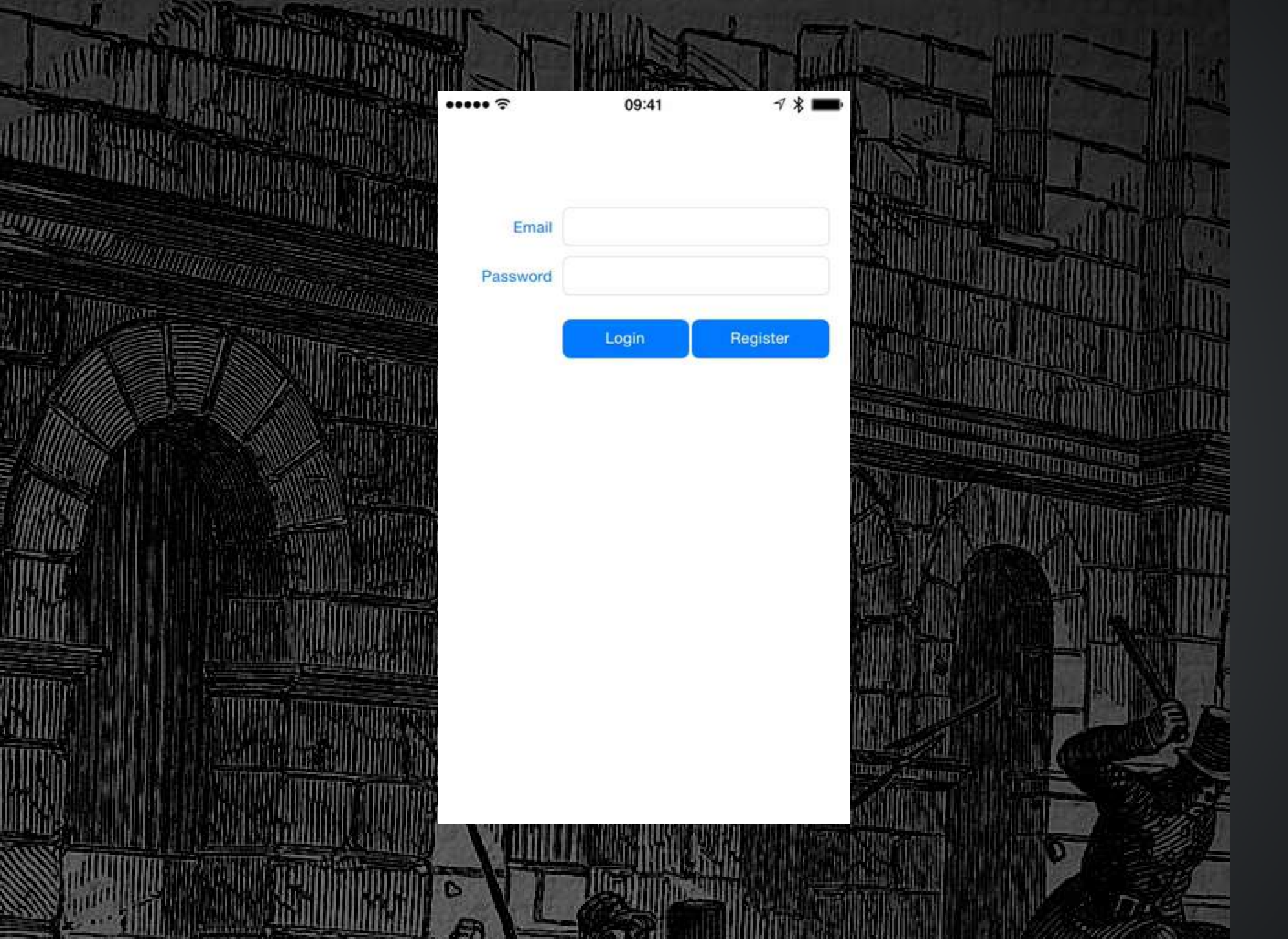
```
val userFilter = new UserFilter()
val normalizer = new ZScoreNormalizer()
val intensityFeatureExtractor =
                    new IntensityFeatureExtrac
val intensityPredictor = new LinearRegressi
    .setLabelCol("label")
    .setFeaturesCol("features")
    .setPredictionCol("predictions")
```

```
class IntensityFeatureExtractor extends Transformer {

  override def transform(dataset: DataFrame,
                         paramMap: ParamMap): DataFrame
    val useHistory = paramMap.get(useHistoryParam).get
    dataset
      .select("sessionProps.intendedIntensity")
      .rdd
      .map(_.getDouble(0))
      .sliding(useHistory + 1)
      .map(x => (x.head, Vectors.dense(x.tail)))
      .toDF("intensityLabel", "intensityFeatures")
  }

  override def transformSchema(schema: StructType,
    paramMap: ParamMap): StructType =
    StructType(Array(
      StructField("intensityLabel", DoubleType, true),
      StructField("intensityFeatures", VectorType.Vecto
}
```

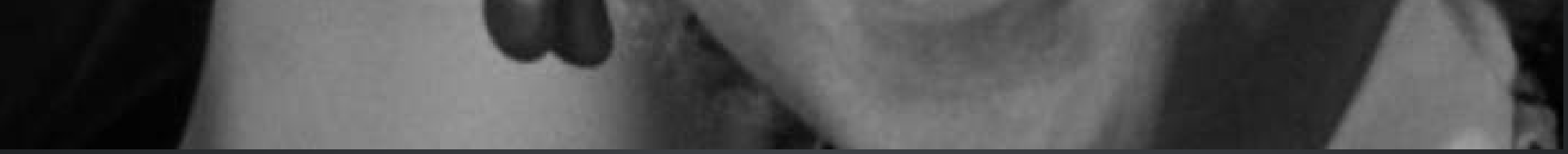

Φ
η
λ

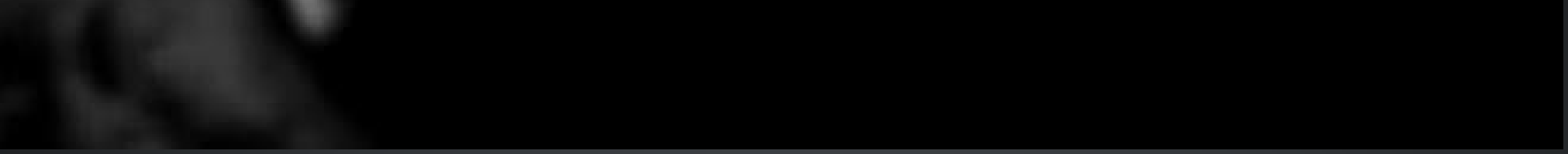
```
implicit val sqlContext = new org.apache.spark.sql.SQL
import sqlContext.implicits._

val events = sc.eventTable().cache().toDF()

val pipeline = new Pipeline().setStages(Array(
  userFilter,
  normalizer,
  intensityFeatureExtractor,
  intensityPredictor
))

getEligibleUsers(events, sessionEndedBefore)
   .map { user =>
     val model = pipeline.fit(
       events,
       ParamMap(ParamPair(userIdParam, user)))
     val testData = //prepare test data
     val predictions = model.transform(testData)
     submitResult(userId, predictions, config)
   }
```


Intensity
Exercise
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2
4
6
8
10
12
14


09:41
Email
Password
Login
Register


SO, TELL ME AGAIN
HOW EVERYTHING'S FINE IN PRODUCTION!

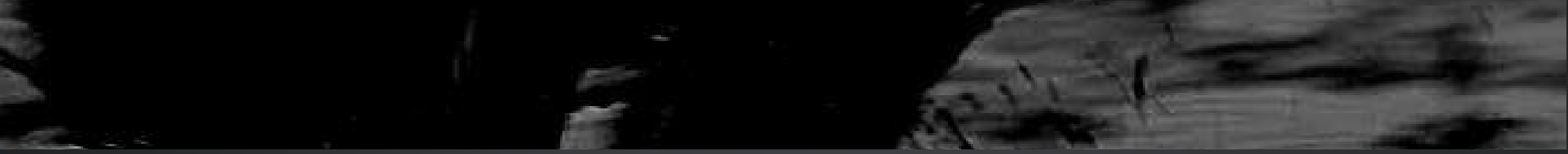


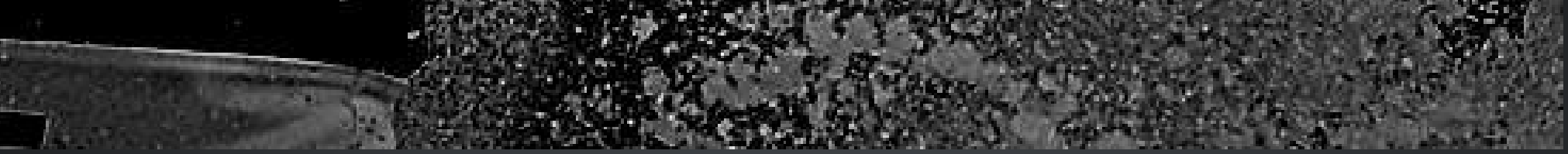


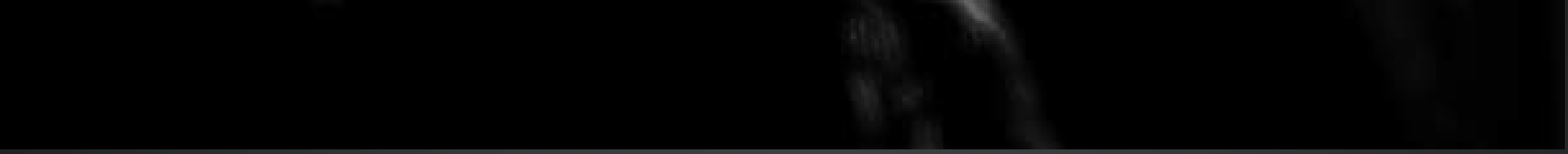
our hardware device vendor for any driver

on:


A (0x8A168D58, 0x88DA1F60, 0xF78C2CBC, 0x000







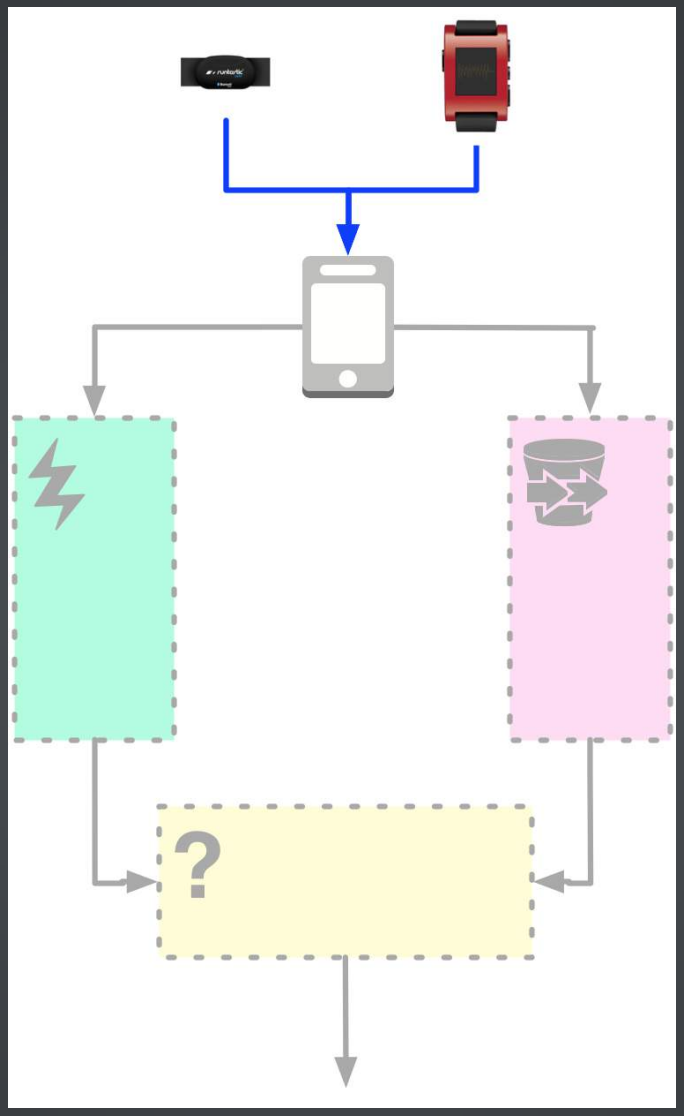
IT'S ALL WRONG

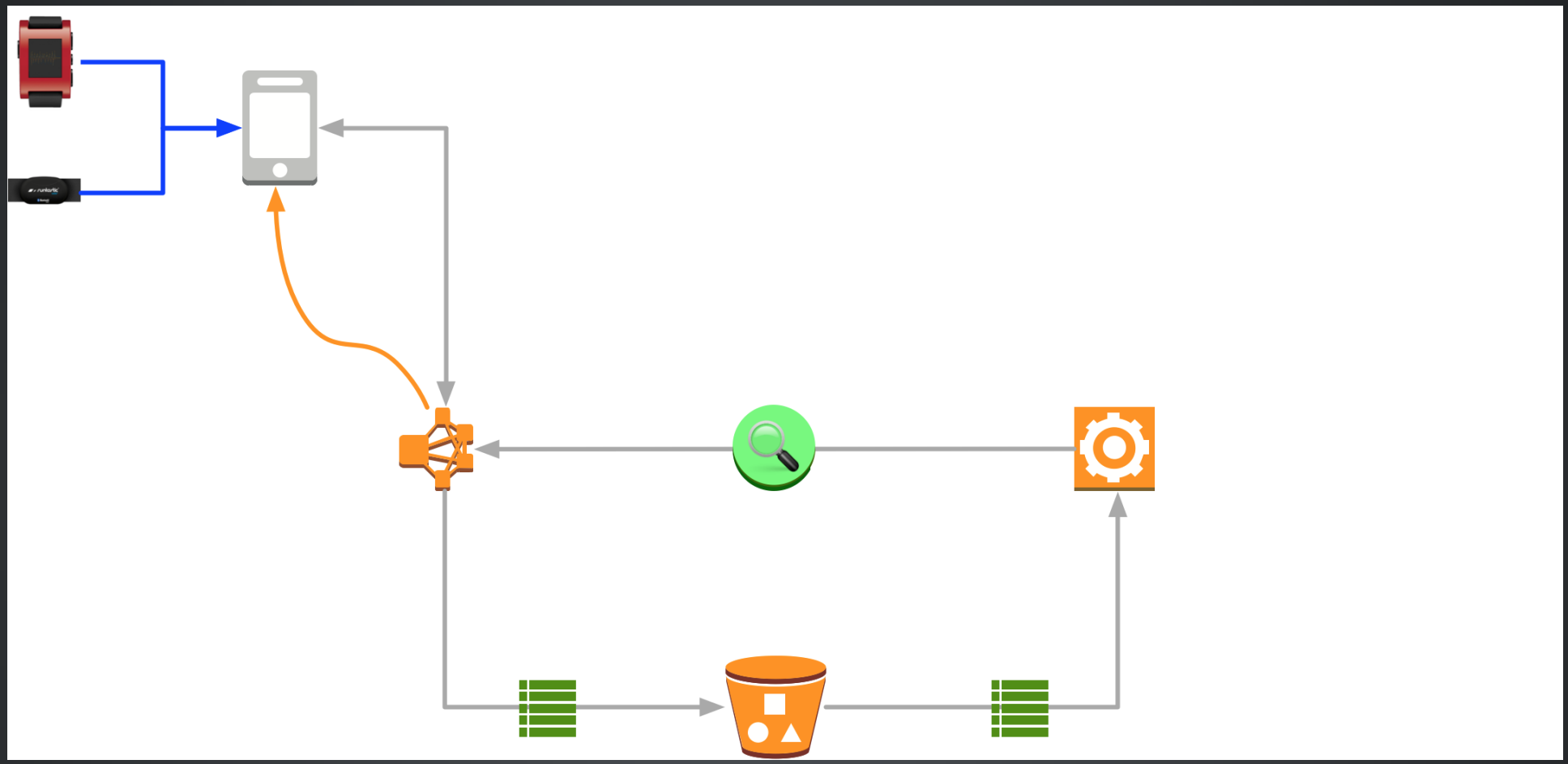
- Data loss from sensors, naïve padding
- Strict network quality requirements
- Large amounts of (unnecessary) data sent on network
- Insufficient feedback on machine learning results

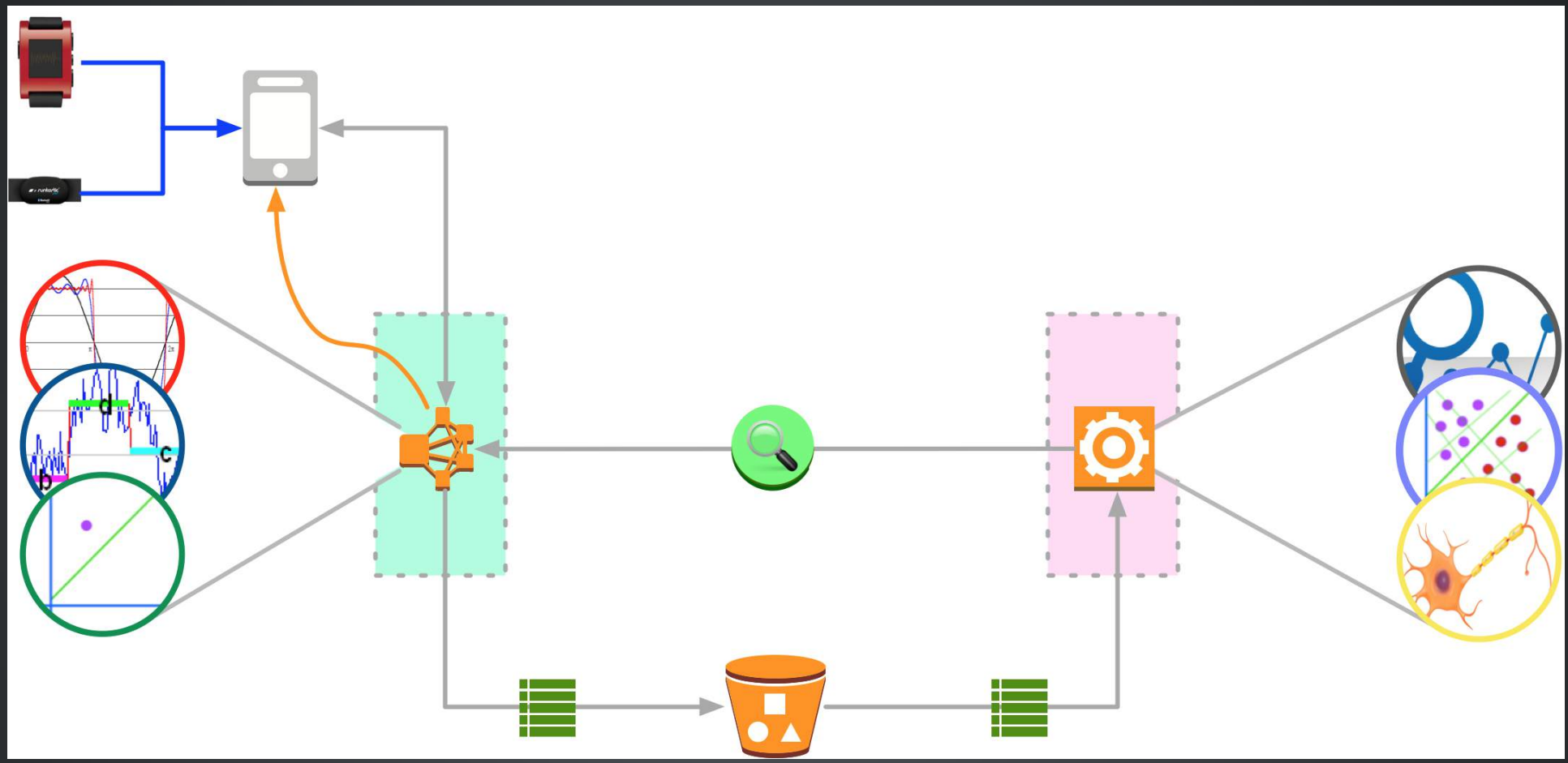
- No timely feedback to the users
- One failing request resulted in offline session
- User experience simply not *slick* enough

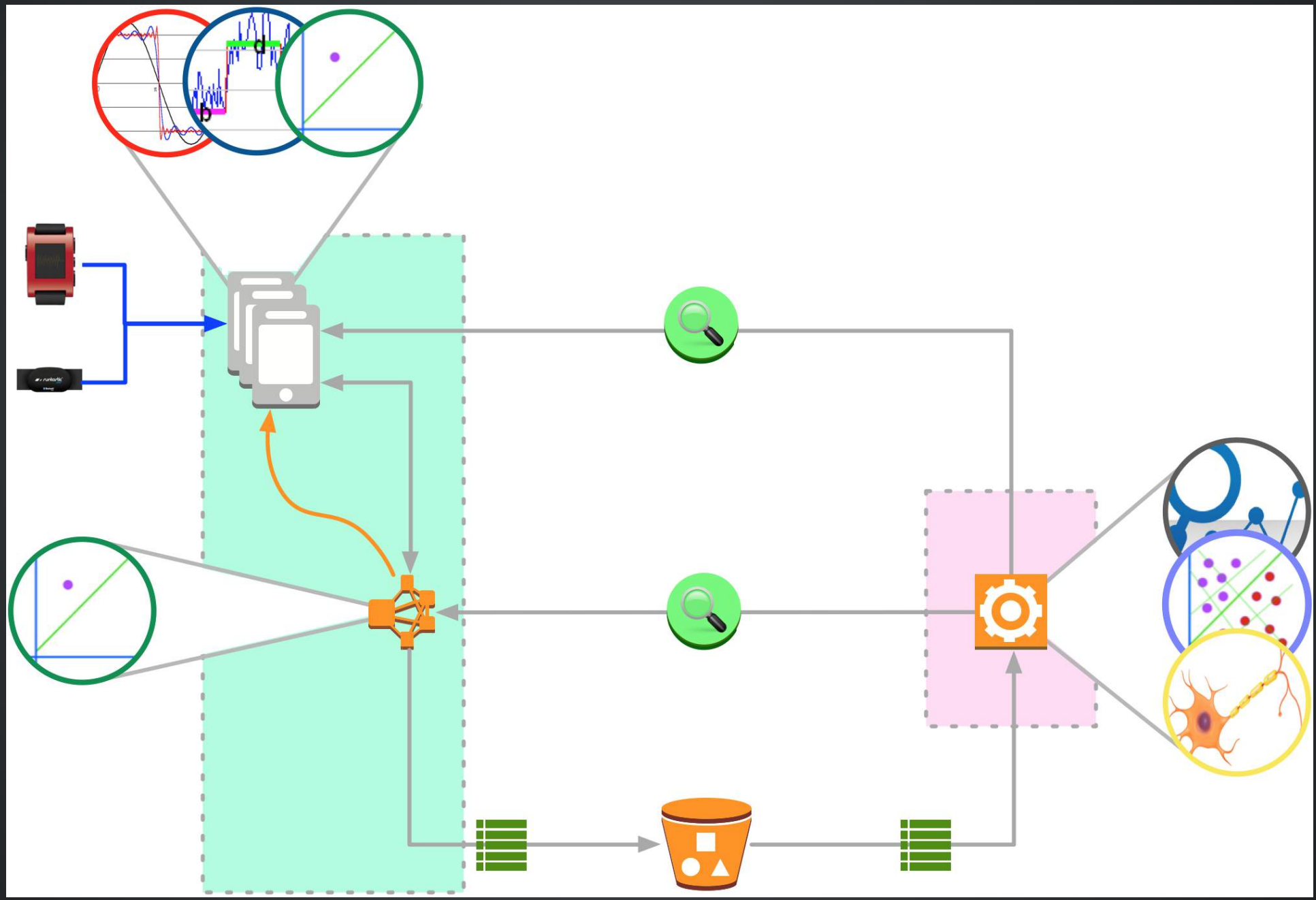
WHAT NOW?

- Better sensor and signal processing
- Shorter experiment cycle
- Distributed lambda architecture with feedback





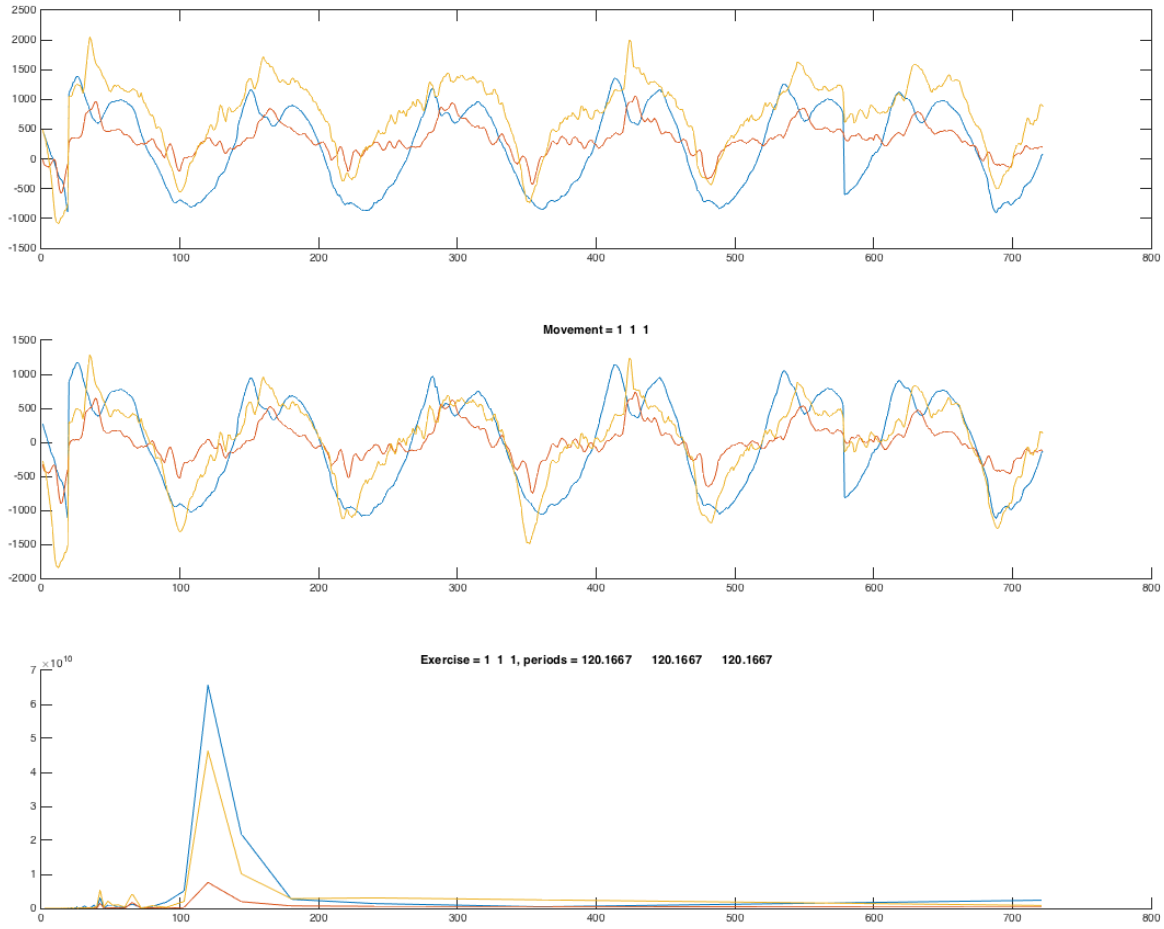

d
c
b



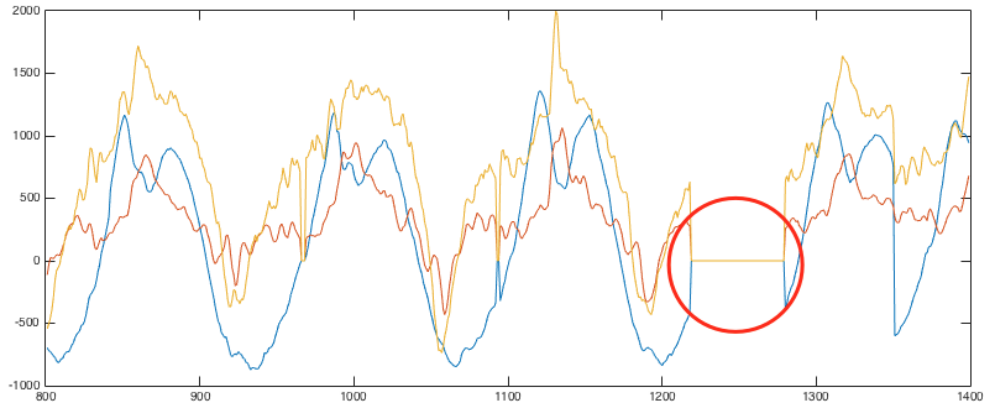
MORE ON THE PHONE

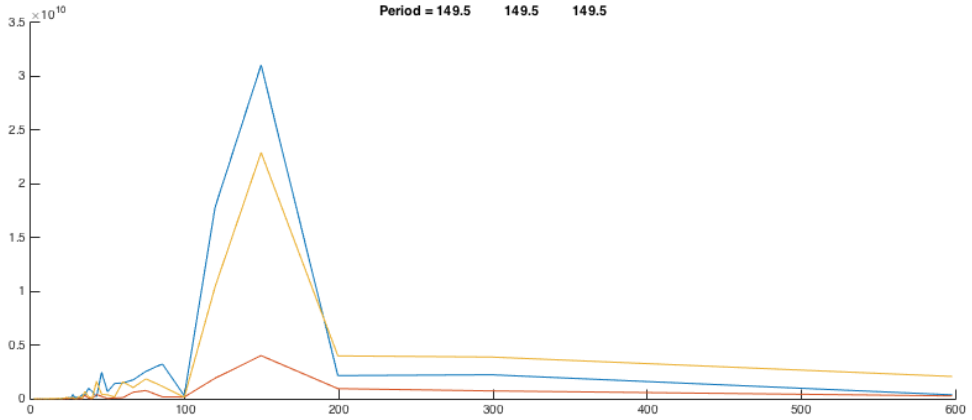
- Responsive
- Resilient

- Core components in C++
- *Multi-platform*: buildable & testable independently
- Smallest possible device-only code
- Sane development tools (e.g. AppCode, CLion, emacs)


Movement = 1 1 1
Exercise = 1 1 1, periods = 120.1667 120.1667 120.1667
×10^10

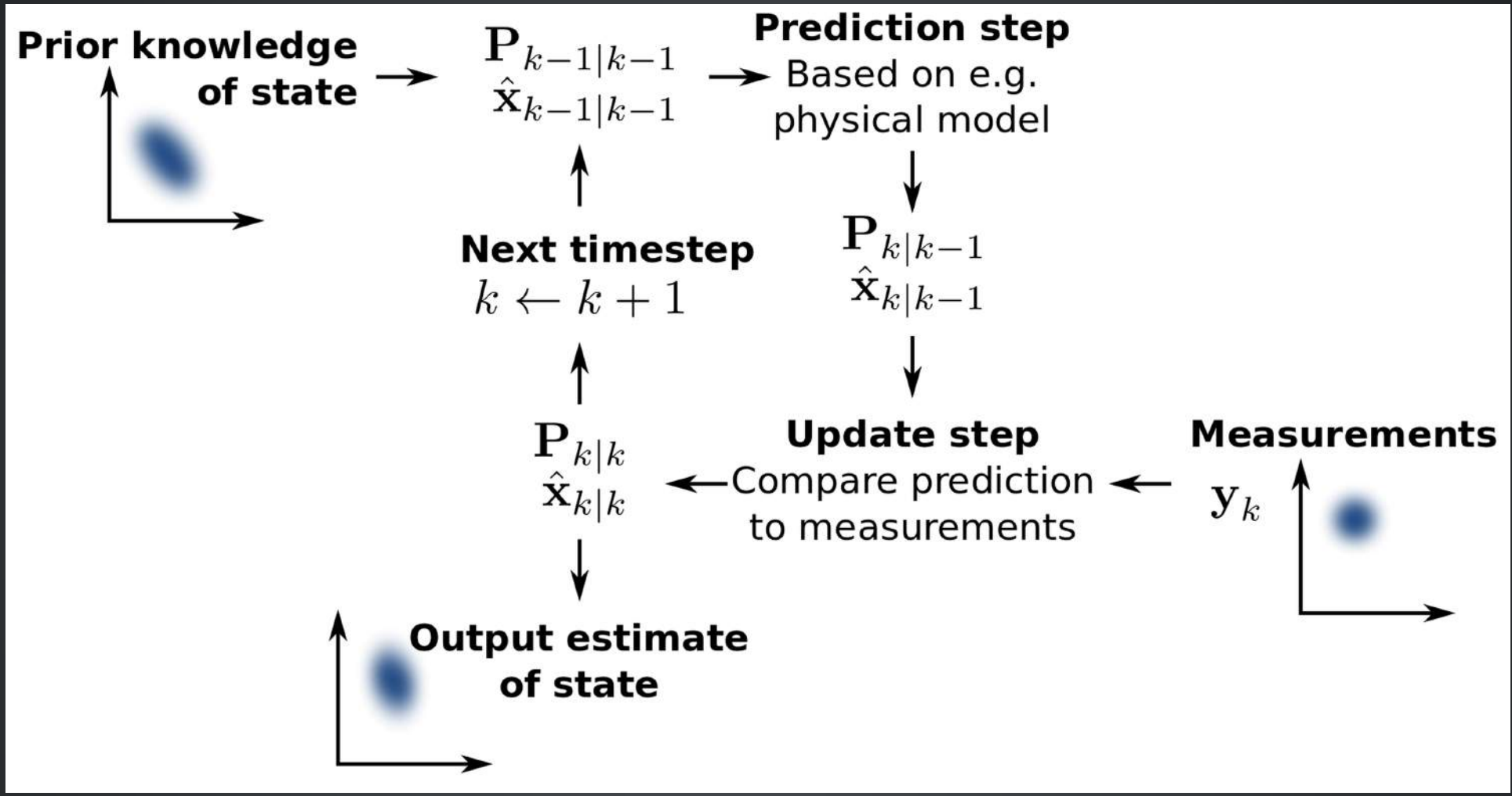
```
function [freq, period, power] = sigfft(xs)
    Y = fft(xs);
    Y(1) = [];
    n = length(Y);
    power = abs(Y(1:floor(n/2))).^2;
    nyquist = 1/2;
    freq = (1:n/2)/(n/2)*nyquist;
    period = 1./freq;
    index = power == max(power);
    freq = period(index);
end

M = readtable('all_4.csv');
ads  = table2array(FM);
time = 1:height(FM);
[freq, period, power] = sigfft(ads);

hold on;
plot(time, ads);
plot(period, power);
hold off;
```


2000
1500
1000
500
0
-500
-1000
800
900
1000
1100
1200
1300
1400


Period = 149.5 149.5 149.5
×10^10
3.5
3
2.5
2
1.5
1
0.5
0
0
100
200
300
400
500
600

$$\ddot{y}(t) = -a$$
$$\dot{y}(t) = \dot{y}(t_0) - a(t - t_0)$$
$$y(t) = y(t_0) + \dot{y}(t_0)(t - t_0) - \frac{a}{2}(t - t_0)^2$$

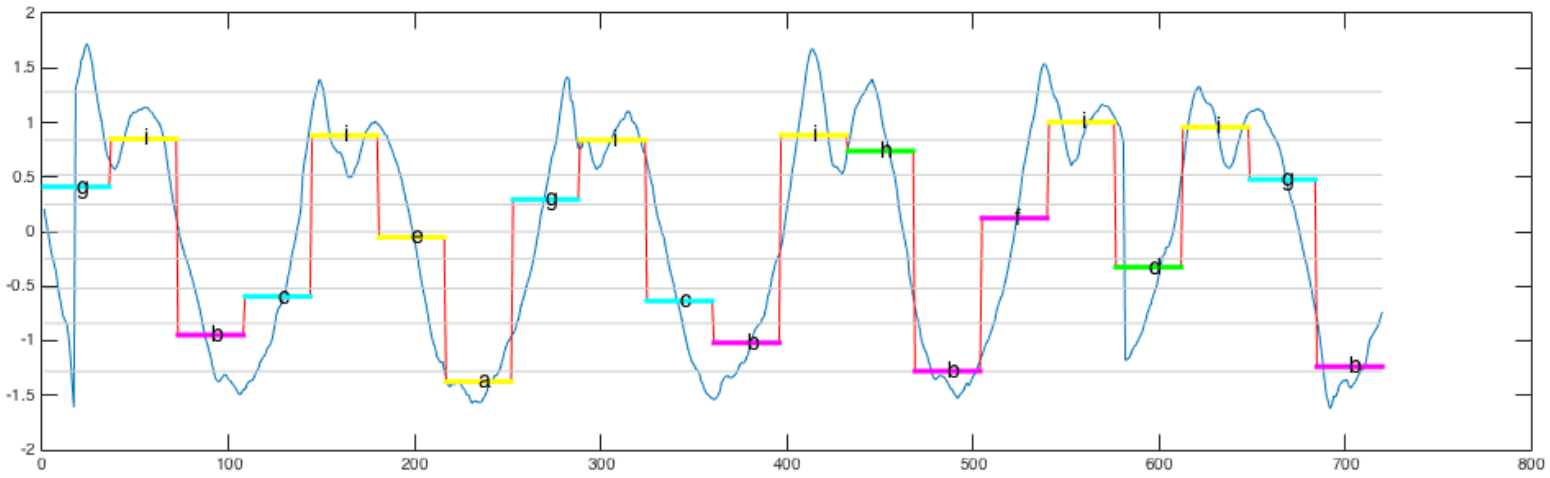
$$y(k+1) = y(k) + \dot{y}(k) - \frac{a}{2}$$

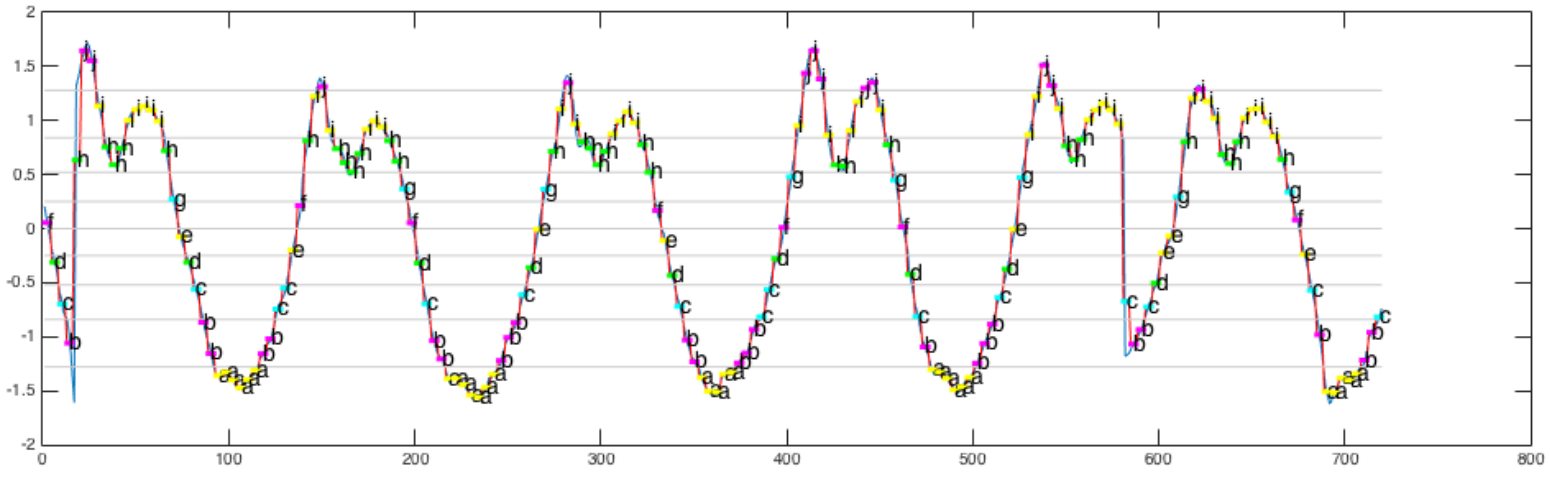

Prior knowledge of state
$\mathbf{P}_{k-1|k-1}$
$\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step
Based on e.g. physical model
$\mathbf{P}_{k|k-1}$
$\hat{\mathbf{x}}_{k|k-1}$
Next timestep
$k \leftarrow k+1$
$\mathbf{P}_{k|k}$
$\hat{\mathbf{x}}_{k|k}$
Update step
Compare prediction to measurements
Measurements
$\mathbf{y}_k$
Output estimate of state

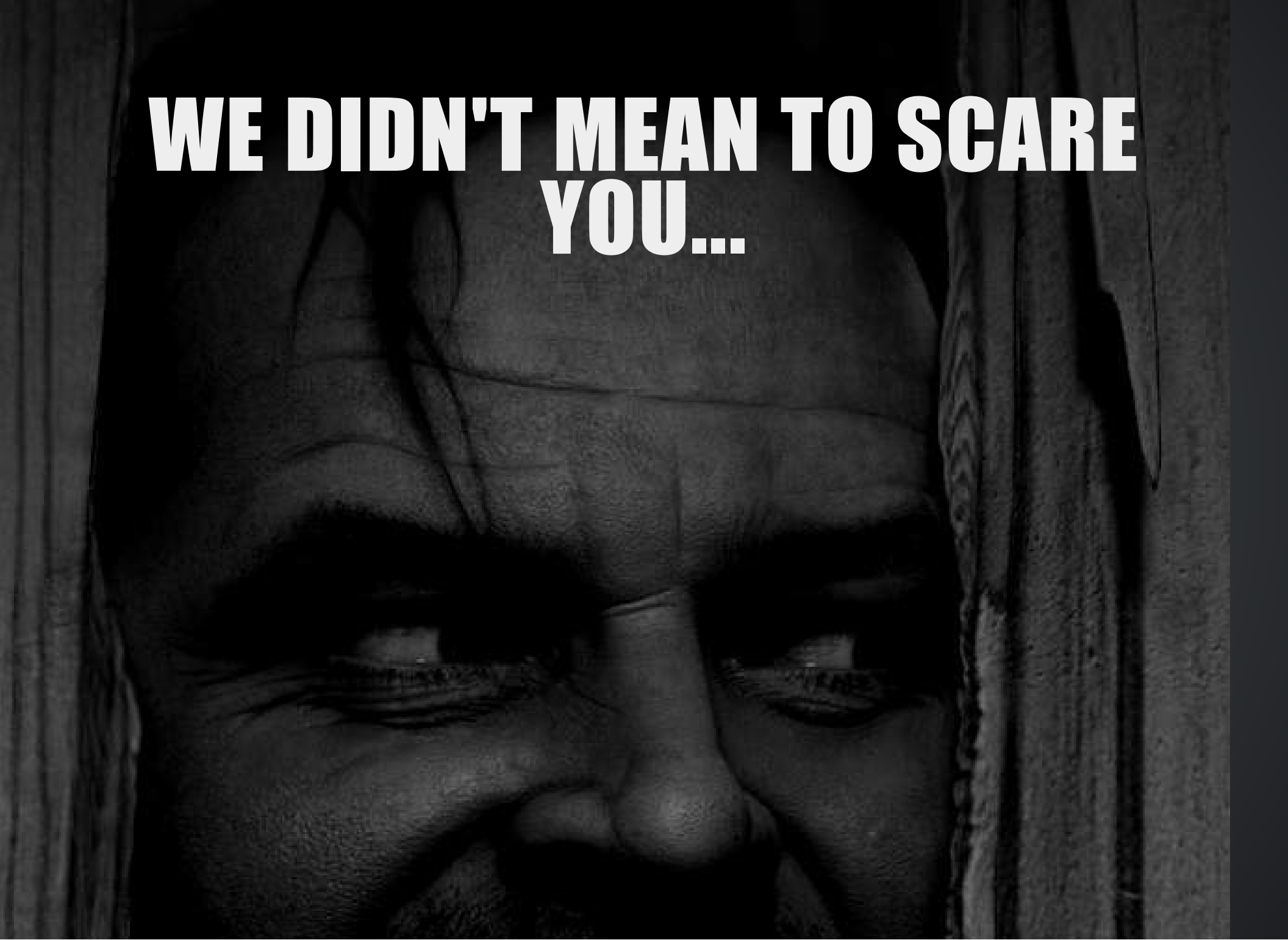
```
Mat result(0, source.cols, source.type(), Scalar(0));
KalmanFilter KF(2, 1, 0);
KF.transitionMatrix = (Mat_<float>(2, 2) << 1, 1, 0, 1);
Mat_<float> measurement(1, 1); measurement.setTo(Scalar(0));

int16_t zero = source.at<int16_t>(0, 0);
KF.statePre.at<float>(0) = 0;
KF.statePre.at<float>(1) = 0;

setIdentity(KF.measurementMatrix);
setIdentity(KF.processNoiseCov, Scalar::all(1));
setIdentity(KF.measurementNoiseCov, Scalar::all(1));
setIdentity(KF.errorCovPost, Scalar::all(2));

Mat col = source.col(0);
for (int j = 0; j < col.rows; ++j) {
  // First predict, to update the internal statePre variable
  KF.predict();

  // The update phase
  measurement(0) = col.at<int16_t>(j);
  float estimate = KF.correct(measurement).at<float>(0);

  ...
}
```

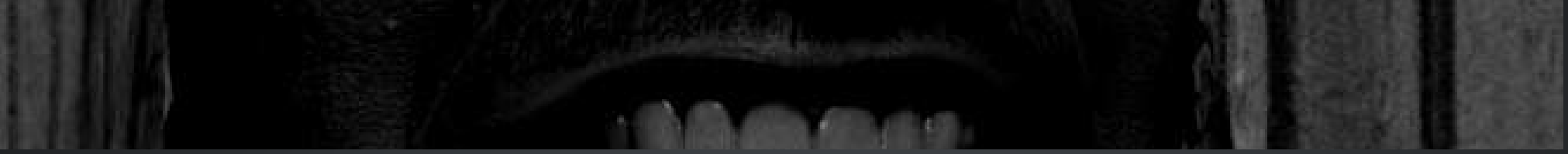
```
exercise_decider::freq_powers exercise_decider::fft(const Mat &source) co
  Mat filtered;
  Mat tmp;
  smooth(source, filtered);
  filtered.convertTo(filtered, CV_32FC1);
  dft(filtered, tmp, DFT_COMPLEX_OUTPUT);
  exercise_decider::freq_powers result(10);
  for (int i = 1; i < tmp.rows / 2; ++i) {
    Complexf v = tmp.at<Complexf>(i, 0);
    exercise_decider::freq_power x { .frequency = tmp.rows / i, .power = p
    result.push_back(x);
  }
  return result;
}

exercise_decider::exercise_result exercise_decider::has_exercise(
  const raw_sensor_data &source, exercise_context &context) const {

  auto pfx = fft(source.data.col(0));
  auto pfy = fft(source.data.col(1));
  auto pfz = fft(source.data.col(2));

  if (!pfx.is_distinct() || !pfy.is_distinct() || !pfz.is_distinct()) retu
  if (!pfx.is_roughly_equal(pfy) || !pfy.is_roughly_equal(pfz)) return no;

  if (context.diverges(pfx, pfy, pfz)) return no;
  context.update(pfx, pfy, pfz);

  return yes;
}
```

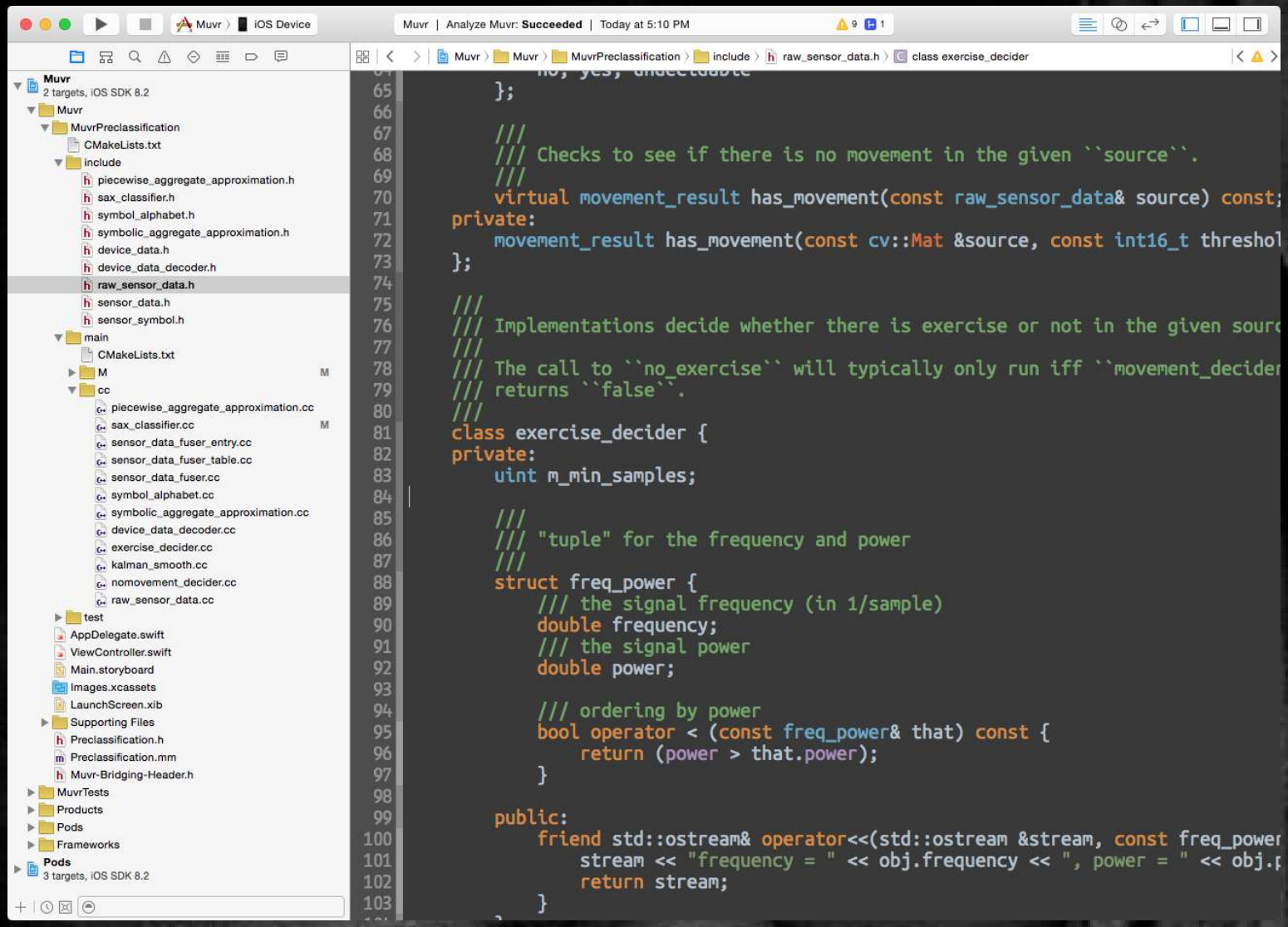
```
raw_sensor_data decode_single_packet(const uint8_t *buffer);

class sensor_data_fuser {
public:
  void push_back(
    const uint8_t *buffer,
    const sensor_location location,
    const sensor_time_t received_at);

  virtual void exercise_block_ended(
    const std::vector<fused_sensor_data> &data,
    const fusion_stats &fusion_stats) = 0;

  virtual void exercise_block_started() = 0;
};

class sax_classifier {
public:
    classification_result classify(const raw_sensor_data &data) const;
    std::vector<char> symbolic_aggregate_approximation(const std::vector<d
                                                       const size_t symbol
    std::vector<double> piecewise_aggregate_approximation(const std::vecto
                                                          const size_t siz
};
```

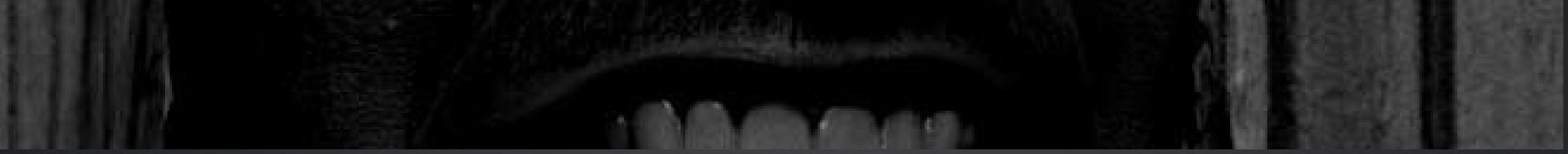

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
100
200
300
400
500
600
700
800


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
100
200
300
400
500
600
700
800


WE DIDN'T MEAN TO SCARE YOU...

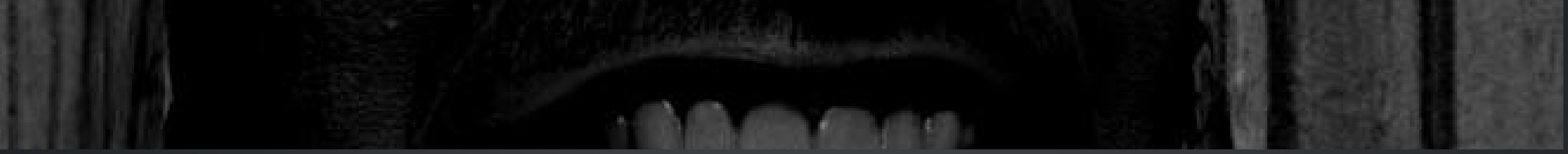



Muvr › iOS Device
Muvr | Analyze Muvr: Succeeded | Today at 5:10 PM
Muvr › Muvr › MuvrPreclassification › include › raw_sensor_data.h › class exercise_decider
Muvr
2 targets, iOS SDK 8.2
Muvr
MuvrPreclassification
CMakeLists.txt
include
piecewise_aggregate_approximation.h
sax_classifier.h
symbol_alphabet.h
symbolic_aggregate_approximation.h
device_data.h
device_data_decoder.h
raw_sensor_data.h
sensor_data.h
sensor_symbol.h
main
CMakeLists.txt
M
cc
piecewise_aggregate_approximation.cc
sax_classifier.cc
sensor_data_fuser_entry.cc
sensor_data_fuser_table.cc
sensor_data_fuser.cc
symbol_alphabet.cc
symbolic_aggregate_approximation.cc
device_data_decoder.cc
exercise_decider.cc
kalman_smooth.cc
nomovement_decider.cc
raw_sensor_data.cc
test
AppDelegate.swift
ViewController.swift
Main.storyboard
Images.xcassets
LaunchScreen.xib
Supporting Files
Preclassification.h
Preclassification.mm
Muvr-Bridging-Header.h
MuvrTests
Products
Pods
Frameworks
Pods
3 targets, iOS SDK 8.2
65 };
66
67 ///
68 /// Checks to see if there is no movement in the given ``source``.
69 ///
70 virtual movement_result has_movement(const raw_sensor_data& source) const;
71 private:
72 movement_result has_movement(const cv::Mat &source, const int16_t threshol
73 };
74
75 ///
76 /// Implementations decide whether there is exercise or not in the given sourc
77 ///
78 /// The call to ``no_exercise`` will typically only run iff ``movement_decider
79 /// returns ``false``.
80 ///
81 class exercise_decider {
82 private:
83 uint m_min_samples;
84
85 ///
86 /// "tuple" for the frequency and power
87 ///
88 struct freq_power {
89 /// the signal frequency (in 1/sample)
90 double frequency;
91 /// the signal power
92 double power;
93
94 /// ordering by power
95 bool operator < (const freq_power& that) const {
96 return (power > that.power);
97 }
98
99 public:
100 friend std::ostream& operator<<(std::ostream &stream, const freq_power
101 stream << "frequency = " << obj.frequency << ", power = " << obj.p
102 return stream;
103 }

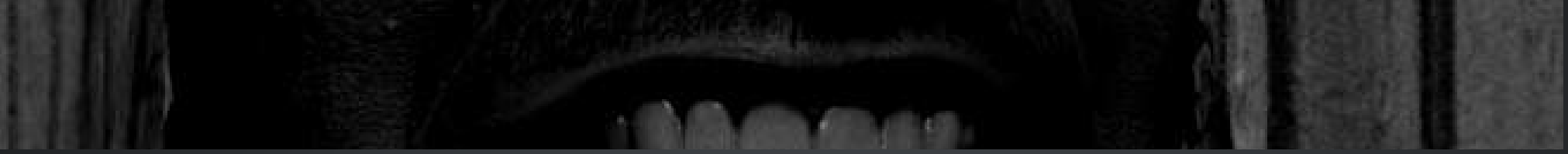


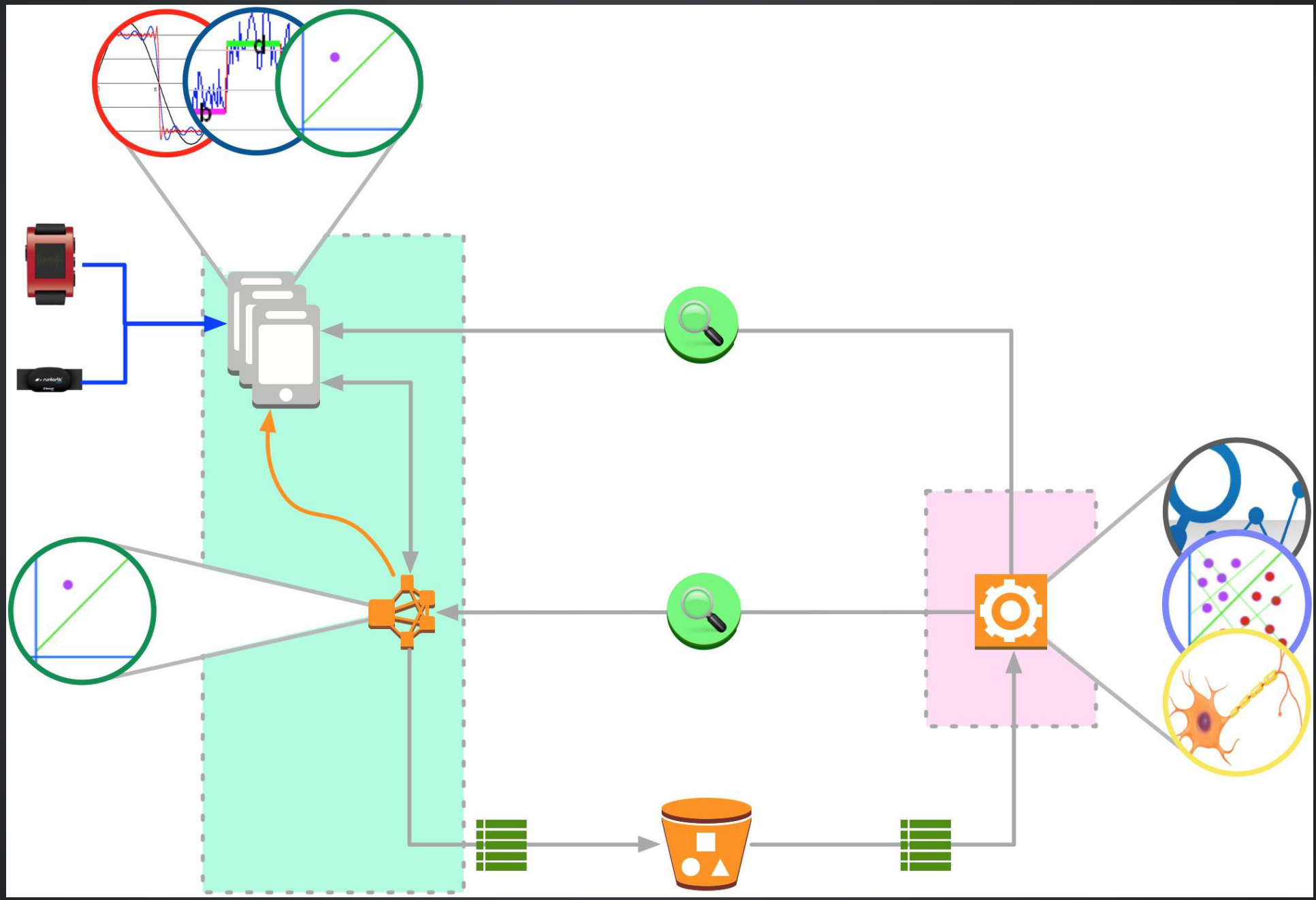
```objc
#import <Foundation/Foundation.h>

@interface Preclassification : NSObject

+ (void)pushBack:(NSData *)data atLocation:(int)locati

@end
```

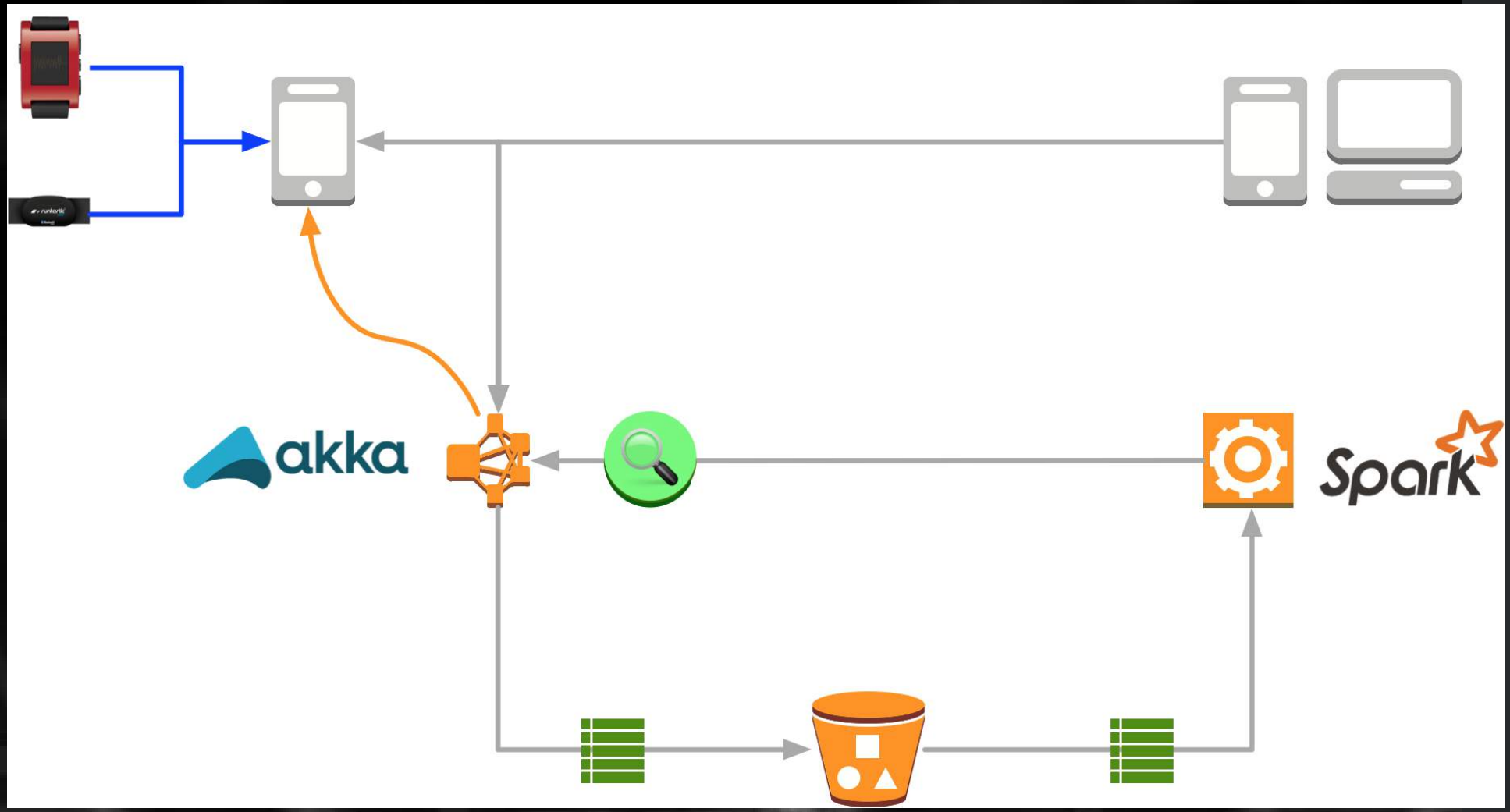
```objc
#import "Preclassification.h"
#import "MuvrPreclassification/include/device_data_dec

@implementation Preclassification

using namespace muvr;

+ (void)pushBack:(NSData *)data atLocation:(int)locati
  const uint8_t *buf = reinterpret_cast<const uint8_t*
  auto res = decode_single_packet(buf);
  ...
}

@end
```

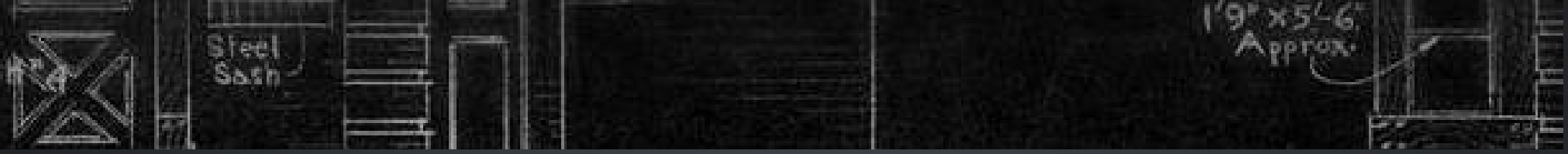


```
class PebbleDeviceSession : DeviceSession {
  private var updateHandler: AnyObject?
  private let adKey   = NSNumber(uint32: 0xface0fb0)
  private let deadKey = NSNumber(uint32: 0x0000dead)

  required init(deviceId: NSUUID, watch: PBWatch!) {
    super.init()
    self.updateHandler = watch.appMessagesAddReceiveUp
      appMessagesReceiveUpdateHandler)
  }

  private func appMessagesReceiveUpdateHandler(watch: 
    data: [NSObject : AnyObject]!) -> Bool {
    if let x = data[adKey] as? NSData {
      Preclassification.pushBack(x, atLocation: .Wrist
    } else if data[deadKey] != nil {
      stop()
    }
    return true
  }
}
```






akka
Spark


Steel
Sash
1'9" x 5'-6"
Approx.

Q&A

THANK YOU!

- ☞ Jobs at www.cakesolutions.net/careers ☜
- Slides at www.eigengo.com/phillyete-2015
- *All* code at github.com/muvr/open-muvr
- Tweets at @honzam399, @zapletal_martin

REFERENCES

- *LTLf and LDLf Monitoring*. Giuseppe De Giacomo, Riccardo De Masellis, Marco Grasso, Fabrizio Maria Maggi and Marco Montali, 2014
- *User Exercise Pattern Prediction through Mobile Sensing*. Georgi Kotsev, Le T. Nguyen, Ming Zeng, and Joy Zhang, 2014
- *Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors*. Ming Zeng, Le T. Nguyen, Bo Yu, Ole J. Mengshoel, Jiang Zhu, Pang Wu and Joy Zhang, 2014
- *Fast Prediction with SVM Models Containing RBF Kernels*. Marc Claesen, Frank De Smet, Johan A.K. Suykens and Bart De Moor, 2014
- *Adaptive Activity Recognition with Dynamic Heterogeneous Sensor Fusion*. Ming Zeng, Xiao Wang, Le T. Nguyen, Pang Wu, Ole J. Mengshoel and Joy Zhang, 2014
- *Parametric Linear Dynamic Logic*. Peter Faymonville and Martin Zimmermann, 2014
- *Time-Series Classification Through Histograms of Symbolic Polynomials*. Josif Grabocka, Martin Wistuba and Lars Schmidt-Thieme, 2013
- *Accelerometer-based Energy Expenditure Estimation Methods and Performance Comparison*. Fang-Chen Chuang, Ya-Ting C. Yang and Jeen-Shing Wang, 2013
- *Linear Temporal Logic and Linear Dynamic Logic on Finite Traces*. Giuseppe De Giacomo and Moshe Y. Vardi, 2013
- *Visualizing Variable-Length Time Series Motifs*. Yuan Li, Jessica Lin and Tim Oates, 2012
- *Segmenting Time Series: A Survey and Novel Approach*. Eamonn Keogh, Selina Chu, David Hart and Michael Pazzani, 2011
- *Finding Motifs in Time Series*. Jessica Lin, Eamonn Keogh, Stefano Lonardi and Pranav Patel, 2002